piXedfit Documentation

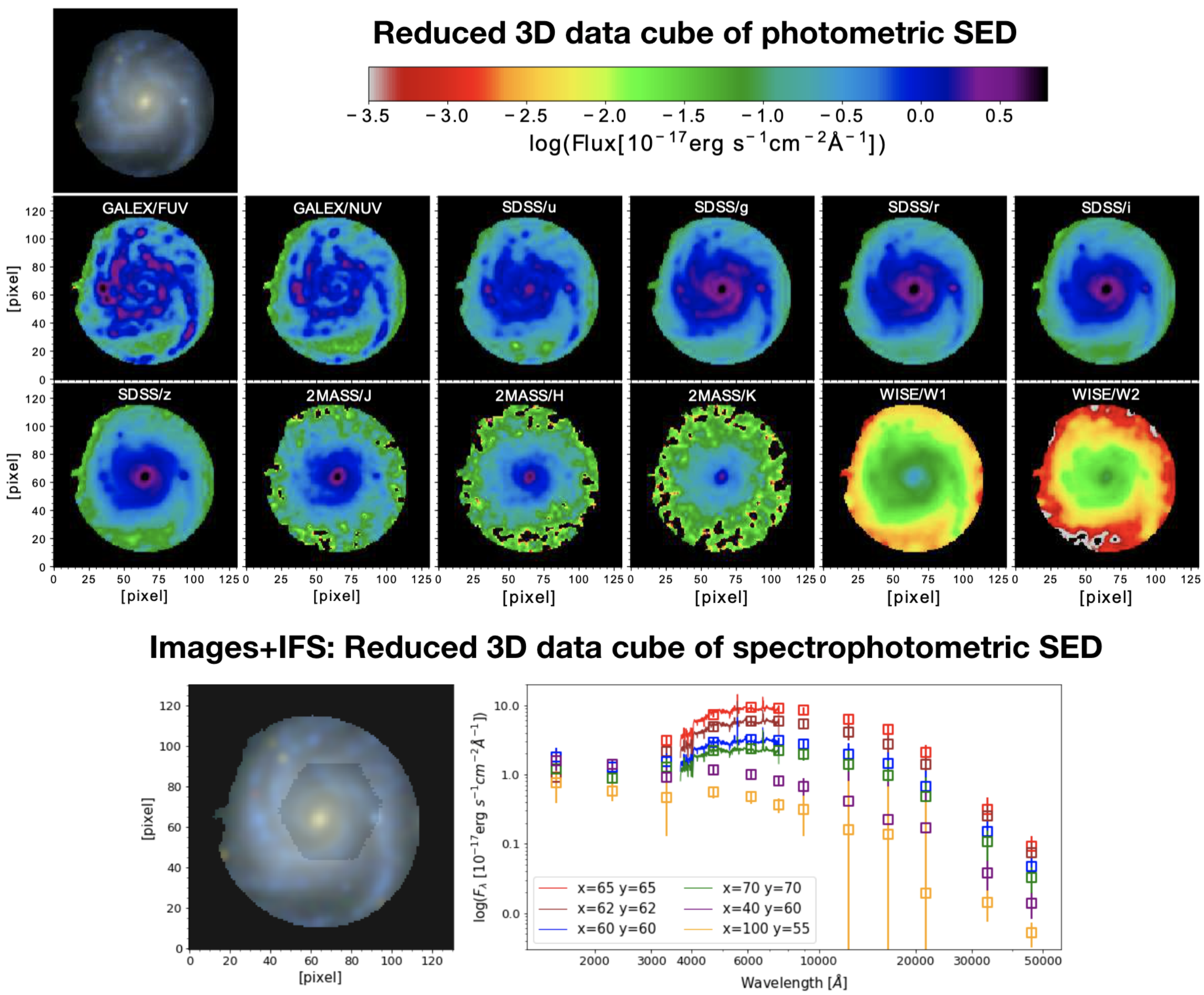
Release 1.0

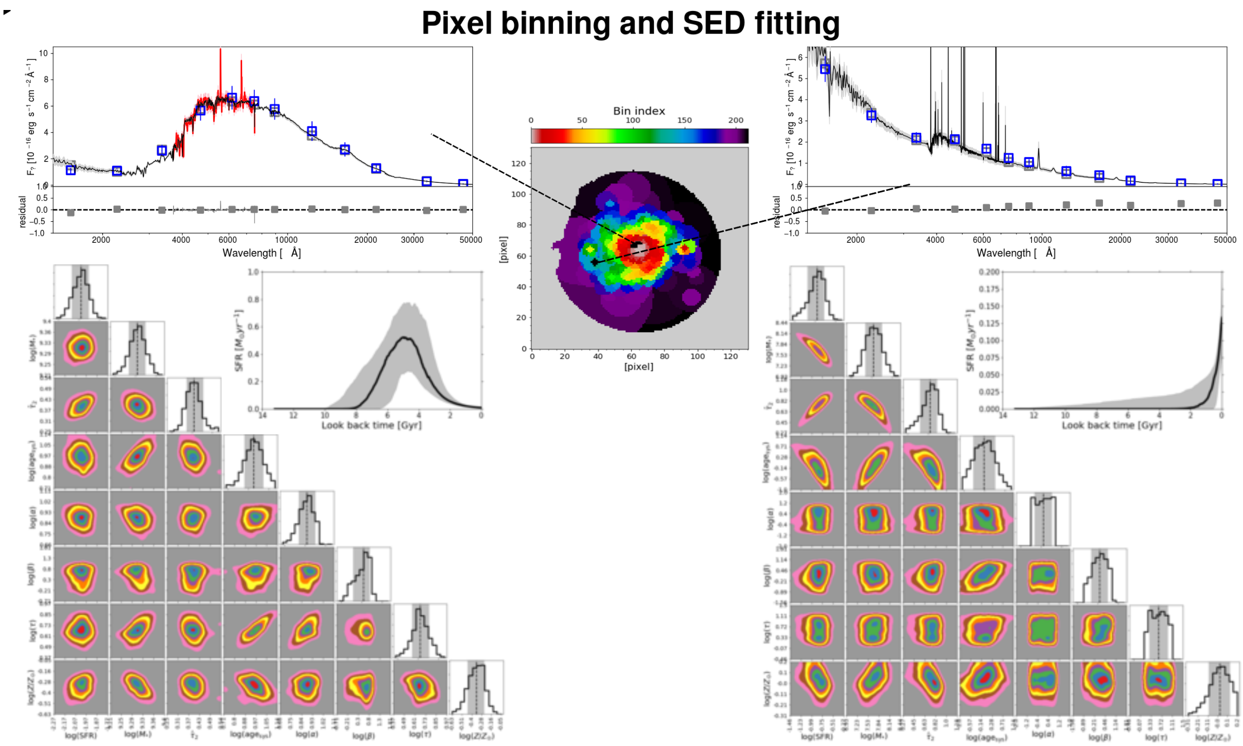
Abdurro'uf

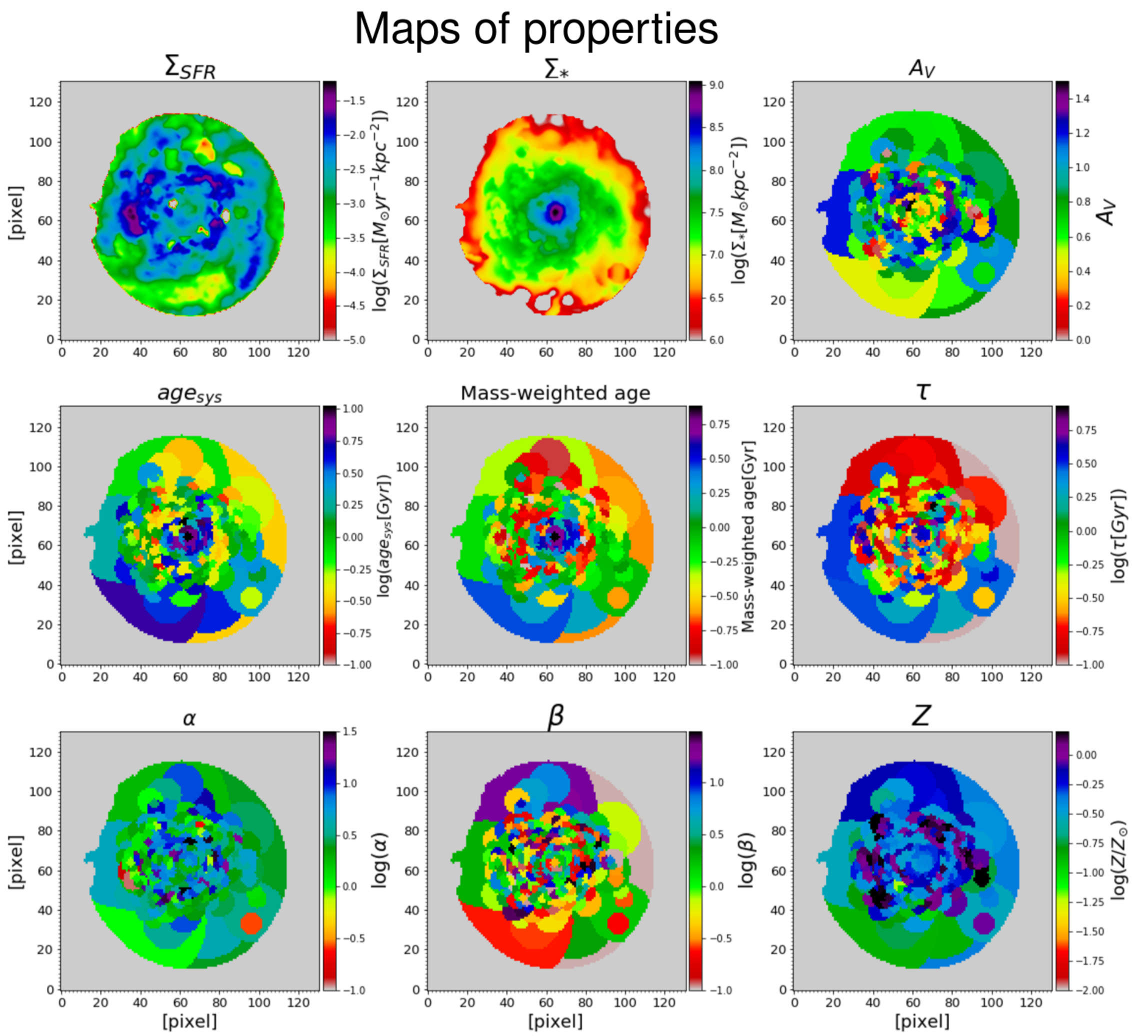
Sep 13, 2023

USER GUIDE

piXefit provides a compehensive set of tools for analyzing spatially resolved spectral energy distributions (SEDs) of galaxies and dissecting the spatially resolved properties of the stellar populations and dust in the galaxies. First, it can produce a pixel-matched 3D data cube from an input of a set of mutliband imaging data alone or in combination with an integral field spectroscopy (IFS) data. When IFS data is provided, it can produce a 3D spectrophotometric data cube in which spectra and photometric SEDs are combined on pixel level. Second, it has a unique pixel binning feature that can optimize the S/N ratio of SEDs on spatially resolved scales while retaining the spatial and spectral variations of the SEDs by accounting the similarity of SED shape of pixels in the binning process. This can be expected to reduce biases introduced by the binning process that combines pixels regardless of the variations in their SED shapes. Finally, piXedfit also provides a stand-alone SED fitting capability. It has two options of fitting methods: MCMC and random dense sampling of parameter space (RDSPS). Most of the modules in piXedfit have implemented MPI for parallel computation. A detailed description of piXedfit is presented in Abdurro'uf et al. (2021). Some examples of practical usages and tutorials can be found at examples.




Pixel binning and SED fitting
Bin index
[pixel]
[pixel]
Wavelength [Å]
Wavelength [Å]
residual
residual
Look back time [Gyr]
Look back time [Gyr]
log(SFR)
log(M*)
log(α)
log(β)
log(τ)
log(Z/Z⊙)


Maps of properties
Σ_{SFR}
Σ_*
A_V
age_{sys}
Mass-weighted age
τ
α
β
Z
[pixel]
$\log(\Sigma_{SFR}[M_\odot yr^{-1} kpc^{-2}])$
$\log(\Sigma_*[M_\odot kpc^{-2}])$
$\log(age_{sys}[Gyr])$
Mass-weighted age[Gyr]
$\log(\tau[Gyr])$
$\log(\alpha)$
$\log(\beta)$
$\log(Z/Z_\odot)$

CHAPTER

ONE

FEATURES

piXedfit has six modules that work independently with each other. For instance, it is possible to use the SED fitting module for fitting either global (integrated) or spatially resolved SEDs of galaxies. Those modules include:

- `piXedfit_images`: image processing.
- `piXedfit_spectrophotometric`: spatial and spectral matching between multiband imaging data and IFS data.
- `piXedfit_bin`: pixel binning to optimize S/N of SEDs on spatially resolved scales.
- `piXedfit_model`: generating model SEDs.
- `piXedfit_fitting`: SED fitting on spatially resolved scales or global (integrated) scales.
- `piXedfit_analysis`: Analysis of SED fitting result, including visualization plots and retrieving best-fitting parameters.

1.1 Installation

1.1.1 Dependencies

- Python >= 3.7
- FSPS and Python FSPS. Note that we need to set an environmental variable called SPS_HOME to make FSPS works. Please follow the instruction in FSPS website for this.
- mpi4py for parallel processing.
- emcee for SED fitting with MCMC method.
- HDF5 and h5py (follow the installation instruction from the website).

1.1.2 Installing piXedfit for the first time

- cd to a desired installation directory, clone **piXedfit**, and install.

```
cd <install_dir>
git clone https://github.com/aabdurrouf/piXedfit.git
cd piXedfit
python -m pip install .
```

- Set an environmental variable called PIXEDFIT_HOME that point to **piXedfit** parent directory.

```
export PIXEDFIT_HOME="$PWD"
```

- The issue with the above command is that we need to do it every time we open a new terminal. Alternatively, we can add this environmental variable to .bashrc (add export PIXEDFIT_HOME="path_to_piXedfit" to the last line in the .bashrc file).

```
vi ~/.bashrc
```

Add the above line and then do

```
source  ~/.bashrc
```

- To add the environmental variable permanently, we can add the same line to .bash_profile or .profile. In case of .bash_profile:

```
vi ~/.bash_profile
```

Add the line and then do

```
source ~/.bash_profile
```

1.1.3 Upgrading piXedfit

- cd to the installation directory, clone **piXedfit**, and install.

```
cd <install_dir>
git clone https://github.com/aabdurrouf/piXedfit.git temp
cp -r temp/piXedfit temp/requirements.txt temp/setup.py piXedfit/
rm -rf temp
cd piXedfit
python -m pip install .
```

1.2 Managing filters

Before proceeding any analysis with **piXedfit**, we should make sure that our photometric filters (i.e., the transmission curves) are recognized by **piXedfit**. For this, **piXedfit** provides functions for managing the library of filters within the **piXedfit** system. Below, we will demonstrate how to manage the filters library. For more information about the functions, please look at the API reference here.

1.2.1 See available filters

To see a list of filters that already available in **piXedfit**, we can use `list_filters()` function.

```
from piXedfit.utils.filtering import list_filters

filters = list_filters()
```

1.2.2 Add filter

We can also add a new filter transmission using add_filter() function.

```
from piXedfit.utils.filtering import add_filter

filter_name = 'name_of_filter'
filter_wave = fil_w                     # wavelength grid in the transmission␣
↪function
filter_transmission = fil_t             # transmission function
filter_cwave = c_wave                   # central wavelength of the filter
add_filter(filter_name, filter_wave, filter_transmission, filter_cwave)
```

1.2.3 Remove filter

To remove a particular filter from the library in **piXedfit**, we can use remove_filter() function.

```
from piXedfit.utils.filtering import remove_filter

remove_filter(filter_name)
```

1.2.4 Change filter name

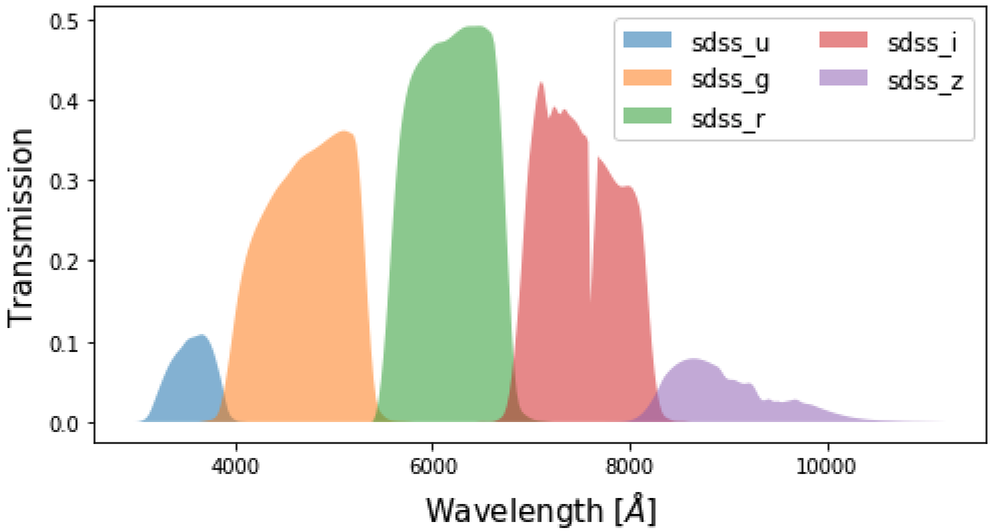
One can change name of a filter in the library using change_filter_name() function.

```
from piXedfit.utils.filtering import change_filter_name

change_filter_name(old_filter_name, new_filter_name)
```

1.2.5 Get filter transmission

It is also possible to get the transmission curve of a filter. The following is an example of script to retrieve and plot the transmision curves of SDSS filters using the get_filter_curve().

```
from piXedfit.utils.filtering import get_filter_curve

filters = ['sdss_u', 'sdss_g', 'sdss_r', 'sdss_i', 'sdss_z']

fig1 = plt.figure(figsize=(8,4))
f1 = plt.subplot()
plt.xlabel(r"Wavelength [$\AA$]", fontsize=15)
plt.ylabel("Transmission", fontsize=15)

for bb in range(0,len(filters)):
    fil_w, fil_t = get_filter_curve(filters[bb])

    f1.fill_between(fil_w, 0, fil_t, alpha=0.5, label=filters[bb])

plt.legend(loc=1, ncol=2, fontsize=12)
```



1.2.6 Get central wavelengths of filters

One can also get the central wavelengths of filters using the `cwave_filters()` function.

```
from piXedfit.utils.filtering import cwave_filters

photo_wave = cwave_filters(filters)
```

`filters` is a list of filter names.

1.3 List of imaging data

The list of imaging dataset that can be analyzed with the current version of **piXedfit** and a brief description on their specifications, unit of pixel value, and how to estimate flux uncertainty are given in the following.

- **Galaxy Evolution Explorer (GALEX)**

  The input image (in FITS) is assumed to have the same format as the one obtained from the MAST. Commonly, the background subtraction has been done for the imaging data product and the background image is provided in a seperate FITS file. The imaging data in the two bands (FUV and NUV) have spatial resolution (i.e., FWHM of PSF) of 4.2" and 5.3", respectively. The spatial sampling is 1.5"/pixel. The 5σ limiting magnitudes in FUV (NUV) of the three surveys modes (AIS, MIS, and DIS) are 19.9 (20.8), 22.6 (22.7), 24.8 (24.4), respectively. Pixel value of the imaging data is in unit of counts (i.e., number of detected photons) per second (CPS). For more information, please refer to Morrissey et al. (2007). To convert the pixel value into flux and estimate flux uncertainty, we follow the relevant information from the literature and the survey's website. To convert from pixel value to flux in unit of erg $\mathrm{s}^{-1}\mathrm{cm}^{-2}\mathrm{\AA}^{-1}$, the following equation is used:

  - **FUV**: flux $= 1.40 \times 10^{-15}$CPS
  - **NUV**: flux $= 2.06 \times 10^{-16}$CPS

  To get flux uncertainty, first, uncertainty of counts is estimated using the following equation:

  - **FUV**: $\mathrm{CPS_{err}} = \frac{\sqrt{\mathrm{CPS}\times\text{exp-time}+(0.050\times\mathrm{CPS}\times\text{exp-time})^2}}{\text{exp-time}}$
  - **NUV**: $\mathrm{CPS_{err}} = \frac{\sqrt{\mathrm{CPS}\times\text{exp-time}+(0.027\times\mathrm{CPS}\times\text{exp-time})^2}}{\text{exp-time}}$

The exp-time is exposure time which can be obained from the FITS header (keyword: EXPTIME). Then flux uncertainty can be calculated using the above equations for converting from counts to flux. The above information is taken from GALEX's website.

- **Sloan Digital Sky Survey (SDSS)**

  The input image (in FITS) is assumed to have the same format as that of the Corrected Frame product of SDSS. For this imaging data product, background subtraction has been done and the background image can be reconstructed from a cruder 2D grid of background image sstored in the HDU2 extension of the FITS file. The spatial sampling of the imaging data in the 5 bands (u, g, r, i, and z) is 0.396"/pixel. The median seeing of all SDSS imaging data is 1.32" in the r-band (see Ross et al. 2011). The SDSS imaging is 95% complete to $u = 22.0$ mag, $g = 22.2$ mag, $r = 22.2$ mag, $i = 21.3$ mag, and $z = 20.5$ mag (Abazajian et al. 2004). The pixel value in the SDSS image is counts in unit of nanomaggy. To convert the pixel value into flux in unit of erg $\mathrm{s}^{-1}\mathrm{cm}^{-2}\text{Å}^{-1}$, the following equation is used:

  $$\text{flux} = \text{counts} \times 3.631 \times 10^{6} \times 2.994 \times 10^{-5} \left(\frac{1}{\lambda_c}\right)^2$$

  where the λ_c is the central wavelength of the photometric band.

  To estimate flux uncertainty of a pixel, first, the uncertainty of counts is calculated using the following equation:

  $$\text{counts}_{\text{err}} = \sqrt{\frac{\left(\frac{counts}{NMGY}\right)+\text{counts}_{\text{sky}}}{\text{gain}} + \text{dark variance}}$$

  with NMGY is a conversion factor from counts to flux in unit of nanomaggy (i.e., nanomaggy per count) and $\text{count}_{\text{sky}}$ is counts associated with the sky background image. The $\text{count}_{\text{sky}}$ at a particular coordinate in the image is obtained from bilinear interpolation to the cruder 2D grids of sky background counts stored in the HDU2 of the FITS file. Gain is a convertion factor from count to the detected number of photo electron and dark variance is an additional source of noise from the read-noise and the noise in the dark current. Values of gain and dark variance vary depending on the camera column (camcol) and the photometric band. Those values can be obtained from this SDSS's web page. After getting the $\text{count}_{\text{err}}$, the flux uncertainty can be calculated using the following equation:

  The above information is obtained from this SDSS's web page.

- **Hubble Space telescope (HST)**

  The HST image has a spatial sampling of 0.06"/pixel. The PSF FWHM varies across photometric bands. The PSF FWHM of F160W band is 0.19". The 5σ limiting magnitude of F160W is 26.4 mag. Pixel value of the HST image is counts per second. To convert the pixel value to flux in unit of erg $\mathrm{s}^{-1}\mathrm{cm}^{-2}\text{Å}^{-1}$, a multiplicative convertion factor can be found in the header of the FIST file (keyword: PHOTFLAM). Flux uncertainty of a pixel can be calculated from the weight image, which commonly provided by various surveys.

- **Two Micron All Sky Survey (2MASS)**

  The input image (in FITS) is assumed to be in the same format as that provided by the NASA/IPAC Infrared Science Archive (IRSA). Commonly, the imaging data from that source is not background-subtracted. The imaging data product has spatial sampling of 1.0"/pixel. The seeing is $\sim 2.5 - 3.5$" (Skrutskie et al. 2006). The point-source sensitivities at signal-to-noise ratio S/N=10 are: 15.8, 15.1, and 14.3 mag for J, H, and $\mathrm{K_S}$, respectively. Pixel value of the 2MASS image is in data-number unit (DN). To convert the pixel value to magnitude, one need a magnitude zero-point which can be obtained from the header of the FITS file (keyword: MAGZP). Then the flux can be calculated using a flux for zero-magnitude zero-point conversion values ($f_{\lambda,\text{zero-mag}}$). The $f_{\lambda,\text{zero-mag}}$ in unit of W $\mathrm{cm}^{-2}\mu\mathrm{m}^{-1}$ for the J, H, and $\mathrm{K_S}$ bands are $3.129 \times 10^{-13} \pm 5.464 \times 10^{-15}$, $1.133 \times 10^{-13} \pm 2.212 \times 10^{-15}$, and $4.283 \times 10^{-14} \pm 8.053 \times 10^{-16}$, respectively (see this web page). A proper conversion factor is then applied to convert the flux in the W $\mathrm{cm}^{-2}\mu\mathrm{m}^{-1}$ to erg $\mathrm{s}^{-1}\mathrm{cm}^{-2}\text{Å}^{-1}$. The flux calibration of 2MASS is described in Cohen et al. (2003). The uncertainty of pixel value in a 2MASS image is estimated following the procedure described in the 2MASS survey's website here (see the handy equations in the botttom of the web page).

- **Wide-field Infrared Survey Explorer (WISE)**

  The input image (in FITS) is assumed to be in the same format as that provided in this IRSA website. Commonly, the imaging data from that source is not background-subtracted. The imaging data in the 4 bands 3.4μm ($W1$), 4.6μm ($W2$), 12μm ($W3$), and 22μm ($W4$) have spatial resolutions of 6.1", 6.4", 6.5", and 12.0", respectively. The spatial sampling of the imaging data in the 4 bands is 1.375"/pixel. WISE achieved 5σ point source sensitivities better than 0.08, 0.11, 1.00, and 6.00 mJy in unconfused regions on the ecliptic in the 4 bands (Wright et al. 2010). The pixel value of a WISE image is DN unit. To convert the pixel value to flux in Jy, one need *DN_to_Jy* conversion factor. The *DN_to_Jy* for the $W1$, $W2$, $W3$, and $W4$ are 1.935×10^{-6}, 2.7048×10^{-6}, 1.8326×10^{-6}, and 5.2269×10^{-5}, respectively. The flux is then converted from Jy to erg $\mathrm{s}^{-1}\mathrm{cm}^{-2}\text{Å}^{-1}$. The WISE image atlas product commonly provides uncertainty image that gives the propagated 1σ uncertainty estimate for each pixel in the corresponding coadded intensity image. For estimating flux uncertainty, a relevant instruction is given in a WISE survey's website here.

- **Spitzer (IRAC and MIPS)**

  The mean FWHMs of the PSFs of the 4 IRAC bands 3.6μm, 4.5μm, 5.8μm, and 8.0μm are 1.66", 1.72", 1.88", and 1.98", respectively (Fazio et al. 2004). The mean FWHMs of the PSFs of the 3 MIPS bands 24μm, 70μm, and 160μm are 6.0", 18.0", and 40", respectively (Rieke et al. 2004). The spatial sampling of IRAC imaging data is 1.2"/pixel, while the spatial sammpling of MIPS imaging data varies across bands: 1.5"/pixel (24μm), 4.5"/pixel (70μm), and 9.0"/pixel (160μm). The 1σ point-source sensitivities (with low background and 100 second time frame) of the 4 IRAC bands are 0.6μJy (3.6μm), 1.2μJy (4.5μm), 8.0μJy (5.8μm), and 9.8μJy (8.0μm) (Fazio et al. 2004). The pre-launch estimate of the 1σ confusion limits of the MIPS bands are $\sim 0.5 - 1.3$mJy (70μm) and $\sim 7.0 - 19.0$mJy (160μm) (Xu et al. 2001 and Dole et al. 2003). Pixel values of the IRAC and MIPS are in unit of Mjy/sr. To convert the pixel value to flux density in erg $\mathrm{s}^{-1}\mathrm{cm}^{-2}\text{Å}^{-1}$, one needs pixel size of the image. For estimating the flux uncertainty of a pixel, we use the uncertainty map (commonly provided by surveys, such as SINGS; Kennicutt et al. 2003) whenever available. If the uncertainty map is not available, the flux uncertainty is assumed to be dominated by the calibration uncertainty. The calibration uncertainty of the 4 bands of IRAC is $\sim 10\%$ (Reach et al. 2005; Munoz-Mateos), whereas that uncertainties for the 3 bands of MIPS are 4% (24μm), 5% (70μm), and 12% (160μm) (Engelbracht et al. 2007; Gordon et al. 2007; Stansberry et al. 2007).

- **Herschel (PACS and SPIRE)**

  The three PACS bands have measured PSF FWHMs of 5.67" (70μm), 7.04" (100μm), and 11.18" (160μm) (Anianp et al. 2011, Geis N. and Lutz D. 2010 PACS ICC Document PICC-ME-TN-029 v2.0, Lutz D. 2010 PACS ICC Document PICC-ME-TN-033, and Müller T. 2010 PACS ICC Document PICC-ME-TN-036 v2.0). The three SPIRE bands have mean PSF FWHMs of 18.1" (250μm), 25.2" (350μm), and 36.6" (500μm). The measured confusion noise levels in the 250μm, 350μm, and 500μm bands are 5.8 mJy, 6.3 mJy, and 6.8 mJy, respectively (Griffin ett al. 2010). The PACS imaging data has pixel value in the unit of Jy/pixel, while the SPIRE imaging data varies depending on the survey from which the data is obtained. The SPIRE imaging data provided by the Key Insights on Nearby Galaxies: A Far-Infrared Survey with Herschel (KINGFISH; Kennicutt et al. 2011) has pixel value in the unit of MJy/sr, whereas the SPIRE imaging data provided by the Very Nearby Galaxy Survey (VNGS; Bendo et al. 2012) has pixel value in the unit of Jy/beam. Based on the SPIRE Observer's Manual, the beam areas in arcsec^2 of the 250μm, 350μm, and 500μm are 426, 771, and 1626, respectively. For estimating the flux uncertainty of a pixel, an uncertainty map (such as that provided by the KINGFISH and VNGS surveys) is used whenever available. Otherwise, the flux uncertainty is estimated by assuming that the flux uncertainty is dominated by the calibration uncertainty. The calibration uncertainty of PACS is $\sim 5\%$ (according to the version 4 of the PACS Observer's Manual), while the calibration uncertainty of the SPIRE is $\sim 7\%$ (see this web page).

1.4 Convolution kernels and PSFs

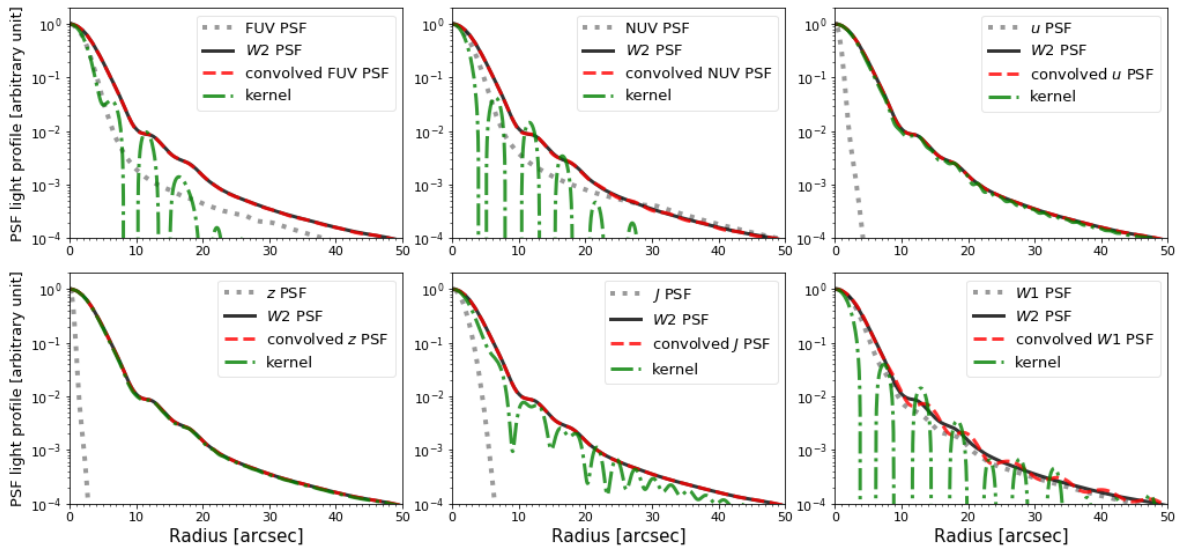
The point spread function (PSF) describes the two-dimensional distribution of light in the telescope focal plane for the astronomical point sources. To get reliable multiwavelength photometric SED from a set of multiband images, especially in the analysis of spatially resolved SEDs of galaxies, it is important to homogenize the spatial resolution (i.e., PSF size) of those images before extracting SEDs from them. This process of homogenizing the PSF size of multiband images is called PSF matching. The final/target spatial resolution to be achieved is the one that is the lowest (i.e., worst) among the multiband images being analyzed. Commonly, PSF matching process of multiband images is done by convolving the images that have higher spatil resolution (i.e., smaller PSF size than the target PSF) with a set of pre-calculated convolution kerenels. The convolution kernel for matching a pair of two PSFs is derived from the ratio of Fourier transforms (e.g., Gordon et al. 2008; Aniano et al. 2011).

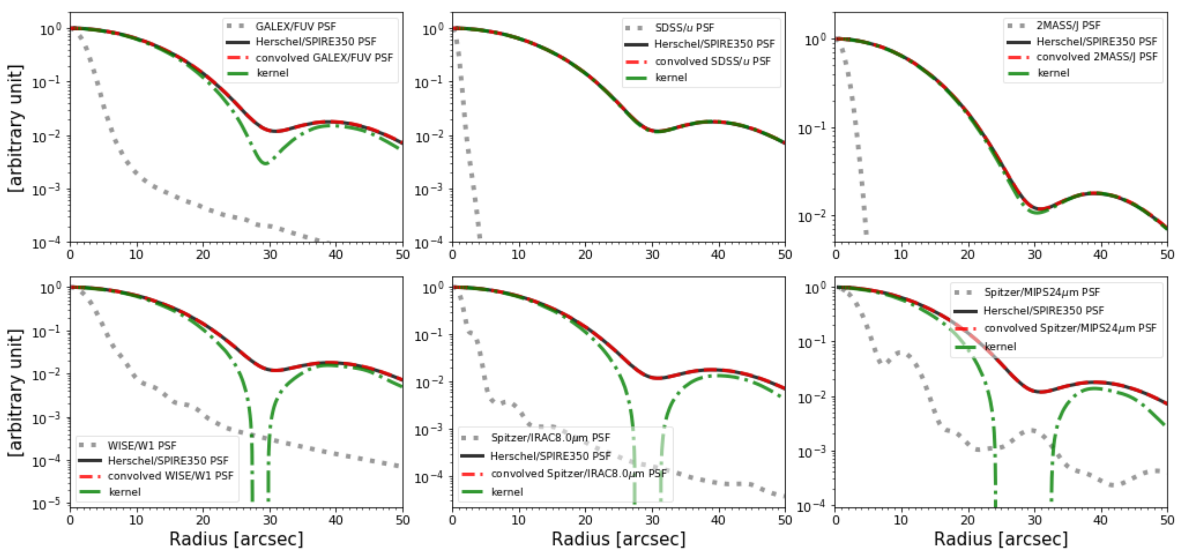
For PSF matching process, `piXedfit_images` module uses convolution kernels from Aniano et al. 2011 that are publicly available at this website. The available kernels cover various ground-based and spaced-based telescopes, including GALEX, Spitzer, WISE, and Herschel. Besides that, Aniano et al. 2011 also provide convolution kernels for some analytical PSFs that includes Gaussian, sum of Gaussians, and Moffat. Those analytical PSFs are expected to be representative of the net (i.e., effective) PSFs of ground-based telescopes.

To associate the PSFs of SDSS and 2MASS (which are not explicitely covered in Aniano et al. 2011) with the analytical PSFs of Aniano et al. 2011, we have constructed empirical PSFs of the 5 SDSS bands and 3 2MASS bands and compare the constructed empirical PSFs with the analytical PSFs from Aniano et al. 2011. We find that the empirical PSFs of SDSS u, g, and r bands are best represented by double Gaussian with FWHM of 1.5", while the other bands (i and z) are best represented by double Gaussian with FWHM of 1.0". For 2MASS, all the 3 bands (J, H, and $\mathrm{K_S}$) are best represented by Gaussian with FWHM of 3.5". Construction of these empirical PSFs is presented in Appendix A of **Abdurro'uf et al. (2020, submitted)**. The empirical PSFs are available at this Github page. For consistency, the `piXedfit_images` use those analytical PSFs to represent the PSFs of SDSS and 2MASS and use the convolution kernels associated with them whenever needed.

By default, when external kernels are not provided by the user, `piXedfit_images` module will use the kernels from Aniano et al. 2011. For flexibility, the users can also input their own kernels to `piXedfit_images`.

Figures below show a demonstration of the performance of some convolution kernels used in the `piXedfit_images` module. In the top figure, the performance of the convolution kernels used to achieve the spatial resolution of WISE/$W2$ is demonstrated. Different panels show different initial PSFs. In the first row from the left to right, we show the convolution results from initial PSFs of GALEX/FUV, GALEX/NUV, and SDSS/u, respectively. The second row, from left to right, we show the result for SDSS/z, 2MASS/J, and 2MASS/$W1$, respectively. In the bottom figure, the performance of the convolution kernels used to achieve the spatial resolution of Herschel/SPIRE350 is demonstrated. In the first row from the left to right, we show the convolution results from initial PSFs of GALEX/FUV, SDSS/u, and 2MASS/J, respectively. The second row, from left to right, we show the result for WISE/$W1$, Spitzer/IRAC 8.0μm, and Spitzer/MIPS 24μm, respectively. The figures shows that the performance of the convolution kernels is very good, evidenced from the good matching between the shapes of the convolved PSFs and the target PSF.





For the characteristic PSFs of the imaging data that can be analyzed with the current version of **piXedfit**, please see the description in this page.

1.5 SED modeling

In **piXedfit**, the task of generating model SEDs is done by `piXedfit_model` module. The SED modeling uses the Flexible Stellar Population Synthesis (FSPS) package through the Python-FSPS as the interface to the Python environment. The FSPS package provides a self-consistent modeling of galaxy's SED through a careful modeling of the physical components that make up the total luminosity output of a galaxy, which consist of stellar emission, nebular emission, dust emission, and emission from the dusty torus heated by the AGN. Since `piXedfit_model` module uses the FSPS model, every parameter (i.e., ingredient) available in the FSPS is also available in the `piXedfit_model`.

1.5.1 SSP model

For modeling a Simple Stellar Population (SSP), the FSPS provides several choices for the Initial Mass Function (IMF), isochrones calculation, and the stellar spectral libraries. The Chabrier et al. (2003) IMF, Padova isochrones (Girardi et al. 2000; Marigo et al. 2007; Marigo et al. 2008), and MILES stellar spectral library (Sanchez-Blazquez et al. 2006; Falcon et al. 2011} are used as the default set in the `piXedfit_model`, but in principle, all the choices available in the FSPS (python-FSPS) are also available in the `piXedfit_model`. In practice, SED fitting procedure demands model SEDs with a random set of Z rather than in a discrete set, as given by the isochrones. In this case, we choose an option in FSPS that allows interpolation of SSP spectra between Z grids. Users of `piXedfit_model` can choose from the 5 available choices of IMF that FSPS provides: Salpeter et al. (1955), Chabrier et al. (2003), Kroupa et al. (2001), van Dokkum et al. (2008), and Dave (2008).

FSPS uses the CLOUDY code (Ferland et al. 1998, 2013) for the nebular emission modeling. The implementation of CLOUDY within FSPS is described in Byler et al. (2017). In short, the modeling has three parameters: SSP age, gas-phase metallicity, and the ionization parameter, U, which represents the ratio of the ionizing photons to the total hydrogen density. By default, the gas-phase metallicity is set to be equal to the model stellar metallicity, and U is fixed to 0.01. The user can also set them as free parameters in the fitting, preferentially if a constraining data is available (e.g., deep optical spectra). The modeling has incorporated the dust attenuation to the emission lines.

There are five options for the dust attenuation modeling in FSPS. We only accommodate two of them in `piXedfit_model`: Calzetti et al. (2000) and the two-component Charlot & Fall (2000) dust attenuation model. In brief, the Calzetti et al. (2000) assumes equal dust attenuation over all starlight regardless of the stellar ages, while Charlot & Fall (2000) assumes an extra attenuation for the light coming from young stars (typically younger than 10 Myr) which still reside in the birth-cloud. For the Calzetti et al. (2000) dust attenuation model, only one parameter is involved, $\hat{\tau}_2$ which represents the dust optical depth. For the two-component Charlot & Fall (2000) model, there are three parameters involved: (1) $\hat{\tau}_1$ controls normalization of the attenuation curve for the birth-cloud component, (2) $\hat{\tau}_2$ controls the normalization of the attenuation curve for the diffuse interstellar medium (ISM) component, and (3) the power-law index n in the dust attenuation curve for the diffuse component (see Eq. 7 and 8 in Leja et al. 2017).

1.5.2 Choices for the SFH

piXedfit adopts the parametric star formation history (SFH) approach, which assumes a functional form for the SFH when generating the model SED of a Composite Stellar Population (CSP). In `piXedfit_model`, there are 5 choices of SFH available:

- **Tau model**

  $SFR(t) \propto e^{-t/\tau}$

  The τ represents the timescale for the declining of the star formation.

- **Delayed tau**

  $SFR(t) \propto te^{-t/\tau}$

  The τ is a parameter that controls the duration of the star formation.

- **Log-normal**

  $SFR(t) \propto \frac{1}{t} \exp \left(-\frac{(\ln(t)-T_0)^2}{2\tau^2} \right)$

  The free parameters T_0 controls the peak location, while τ controls the duration of the star formation.

- **Gaussian**

  $SFR(t) \propto \exp \left(-\frac{(t-T_0)^2}{2\tau^2} \right)$

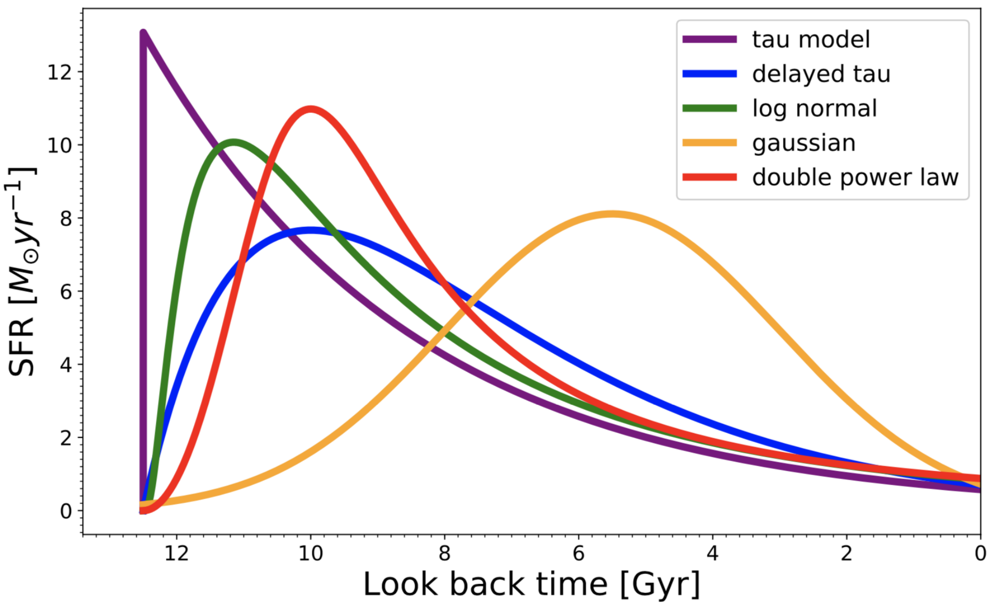
  The T_0 represents the time when star formation reaches the peak, while the τ controls the duration of the star formation.

- **Double power law**

  $SFR(t) \propto \left[\left(\frac{t}{\tau} \right)^{\alpha} + \left(\frac{t}{\tau} \right)^{-\beta} \right]^{-1}$

  The α and β are the falling slope, and the rising slope, respectively. The τ parameter controls the peak time.

All the t in the above equations represent the time since the start of star formation (i.e., age of the system, $\mathrm{age}_{\mathrm{sys}}$). The following figure shows examples of SFHs formed with the 5 SFH choices. All the model SFHs have the same age t of 12.5 Gyr and $M_{*} = 5.0 \times 10^{10} M_{\odot}$. The other SFH parameters are: tau model [$\tau = 4.0$ Gyr], delayed tau [$\tau = 2.5$ Gyr], log-normal [$\tau = 1.0$ Gyr, $T_0 = 1.3$ Gyr], Gaussian [$\tau = 2.5$ Gyr, $T_0 = 7.0$ Gyr], and double power law [$\tau = 2.5$ Gyr, $\alpha = 2.0$ Gyr, $\beta = 2.0$ Gyr].



1.5.3 Dust emission and AGN components

The dust emission modeling in FSPS assumes the energy balance principle, where the amount of energy attenuated by the dust is equal to the amount of energy re-emitted in the infrared (IR) (da Cunha et al. 2008). FSPS uses the Draine & Li (2007) dust emission templates to describe the shape of the IR SED. There are three parameters in the dust emission modeling: U_{min}, γ_e, and Q_{PAH}. U_{min} represents the minimum starlight intensity that illuminate the dust. This minimum starlight intensity is typically found in the diffuse ISM. γ_e represents the fraction of dust mass that is exposed to this minimum starlight intensity. Q_{PAH} represents the fraction of total dust mass that is in the polycyclic aromatic hydrocarbons (PAHs).

For the modeling of emission from the dusty torus heated by the AGN, FSPS uses AGN templates from the Nenkova et al (2008a, b) CLUMPY models. The CLUMPY model uses radiative transfer techniques to approximate the SED from the clumpy dust torus medium which is illuminated by light from the AGN with a broken power-law spectrum. The CLUMPY AGN model is included in the FSPS based on some assumptions that are described in Leja et al. (2018). The modeling has two free parameters: f_{AGN} which represents the total luminosity of the AGN, expressed as a fraction of the galaxy bolometric luminosity, and τ_{AGN} which represents the optical depth of an individual dust clump at 5500 Angstrom in the dusty torus.

1.5.4 IGM absoption, redshifting, and convolving with filters

The `piXedfit_model` has two options for the IGM absorption: Madau (1995) and Inoue et al. (2014). After applying the IGM absorption, the effect of cosmological redshifting and dimming is then applied to the model spectra. After this process, the spectra is transformed into the observer frame flux density (f_λ). Typically, this calculation requires redshift information of the galaxy. Whenever provided, input redshift (if spectroscopic redshift is available) is used. Otherwise, redshift is set as a free parameter in the fitting. The calculation of the luminosity distance uses the cosmology package in the Astropy. The final step in generating model photometric SEDs is convolving the model spectra with the set of filter transmission functions. The current vesion of **piXedfit** has 163 photometric filters of ground-based and space-based telescopes. The user can also add a filter transmission function using `add_filter()` function in `filtering` module.

1.6 Generating model SEDs

piXedfit uses the FSPS for modeling the SED of galaxies. With the Python bindings via Python FSPS, generating model SEDs can be done on-the-fly during the SED fitting process. However, some tasks require a generation of model spectra in a fast pace which turn out to be difficult to achieve. These tasks include the generation of model SEDs that will be used in the spectral matching between the Imaging and IFS data, SED fitting with the random dense sampling of parameter space (RDSPS), and the initial fitting (i.e., burning up) before running the SED fitting with the MCMC method. Please note that the MCMC fitting always uses on-the-fly generation of model SEDs.

For that reason, piXedfit provides an option of generating a set of model spectra at rest-frame. The models are stored in HDF5 file format. The model spectra can be generated using function *`piXedfit.piXedfit_model.save_models_rest_spec()`*. Please see the API reference here for more detailed information about this function. In practice, user only need to generate this set of models once, then these models can be used for various further analyses to multiple galaxies.

1.6.1 Generate random model spectra at rest-frame

To generate random model spectra at rest-frame, we can make a script as shown in the following. You can adjust the model configurations depending on the kind of models you need in your analysis. The ranges of parameters can also be adjusted. Please see the API reference of this function here. The values of parameters are drawn randomly but designed to be uniformly distributed within desired ranges that can be set in the inputs.

```
from piXedfit.piXedfit_model import save_models_rest_spec

imf_type = 1                    # Chabrier (2003)
sfh_form = 4                    # double power law SFH form
dust_law = 0                    # Charlot & Fall (2000) dust attenuation law
duste_switch = 0                # turn off dust emission
add_neb_emission = 1            # turn on nebular emission
add_agn = 0                     # turn off AGN dusty torus emission

nmodels = 100000                # number of model spectra to be generated
nproc = 20                      # number of processors to be used in the␣
↪calculation

# define ranges of some parameters
params_range = {'log_age':[-1.0,1.14], 'dust1':[0.0,3.0], 'dust2':[0.0,3.0]}

name_out = 'model_rest_spec.hdf5'       # name of the output HDF5 file
```

(continues on next page)

(continued from previous page)

```
save_models_rest_spec(imf_type=imf_type, sfh_form=sfh_form, dust_ext_law=dust_
→ext_law,
                        duste_switch=duste_switch, add_neb_emission=add_neb_
→emission,
                        add_agn=add_agn,nmodels=nmodels,nproc=nproc, name_
→out=name_out)
```

The produced models will be used as input in various tasks, including a matching between the imaging and IFS data (see *piXedfit.piXedfit_spectrophotometric.match_imgifs_spectral()* and this example), SED fitting with RDSPS method, and initial fitting in the SED fitting with the MCMC method (see API reference of the *piXedfit_fitting* module).

Note: It is important to note that all the configurations in the SED modeling and SED fitting are determined in this step.

Here we determine what types of models that we want to generate and fit to our observed SEDs. This modeling configuration include the choice of initial mass function (IMF; *imf_type*), the choice of star formation history (SFH; *sfh_form*), the choice of dust attenuation law (*dust_law*), whether to switch on/off the following features (nebular emission *add_neb_emission*, dust emission *duste_switch*, AGN dusty torus emission *add_agn*, intergalactic medium *add_igm_absorption*), and the choice of cosmological parameters. Those features have parameters associated with them. This determines the free parameters that will be involved in the SED fitting process.

Note: For the stellar age parameter (*log_age*) to be sufficiently sampled, it is recommended to set a range for *log_age* with minimum value of -1.0 or -2.0 and maximum value that corresponds to the age of the universe at the redshift of the target galaxy.

1.6.2 Generate random model photometric SEDs at observer-frame

piXedfit also provides a function for generating a set of model photometric SEDs that are calculated at a given redshift. The models are stored in a FITS file format. This kind of data is not requested as input in most of subsequent analyses. Therefore, this functionality is just a complement to other features that already available in piXedfit.

To generate random model photometric SEDs at observer-frame, we can make a script as shown in the following example.

```
from piXedfit.piXedfit_model import save_models_photo

# set of photometric filters
filters = ['galex_fuv', 'galex_nuv', 'sdss_u', 'sdss_g', 'sdss_r', 'sdss_i',
        'sdss_z', '2mass_j', '2mass_h', '2mass_k', 'wise_w1', 'wise_w2',
        'wise_w3', 'wise_w4', 'spitzer_irac_36', 'spitzer_irac_45', 'spitzer_
→irac_58',
        'spitzer_irac_80', 'spitzer_mips_24',  'herschel_pacs_70', 'herschel_
→pacs_100',
        'herschel_pacs_160', 'herschel_spire_250', 'herschel_spire_350']

imf_type = 1                    # Chabrier (2003)
sfh_form = 4                    # double power law SFH form
dust_law = 0                    # Charlot & Fall (2000) dust attenuation law
```

(continues on next page)

(continued from previous page)

```
duste_switch = 1                # turn on dust emission
add_neb_emission = 1            # turn on nebular emission
add_agn = 1                     # turn on AGN dusty torus emission
add_igm_absorption = 0          # turn off absoption effect by the␣
↪intergalactic medium

# cosmology parameters
cosmo = 0                       # Flat LCDM
H0 = 70.0
Om0 = 0.3

nmodels = 100000                # number of model spectra to be generated
nproc = 20                      # number of processors to be used in the␣
↪calculation

gal_z = 0.01

name_out_fits = 'model_photo_seds.fits'
save_models_photo(filters=filters, gal_z=gal_z, imf_type=imf_type, sfh_
↪form=sfh_form,
                dust_ext_law=dust_ext_law, add_igm_absorption=add_igm_
↪absorption,
                duste_switch=duste_switch, add_neb_emission=add_neb_emission,
                add_agn=add_agn, nmodels=nmodels, nproc=nproc, cosmo=cosmo,
                H0=H0, Om0=Om0, name_out_fits=name_out_fits)
```

1.7 Image processing

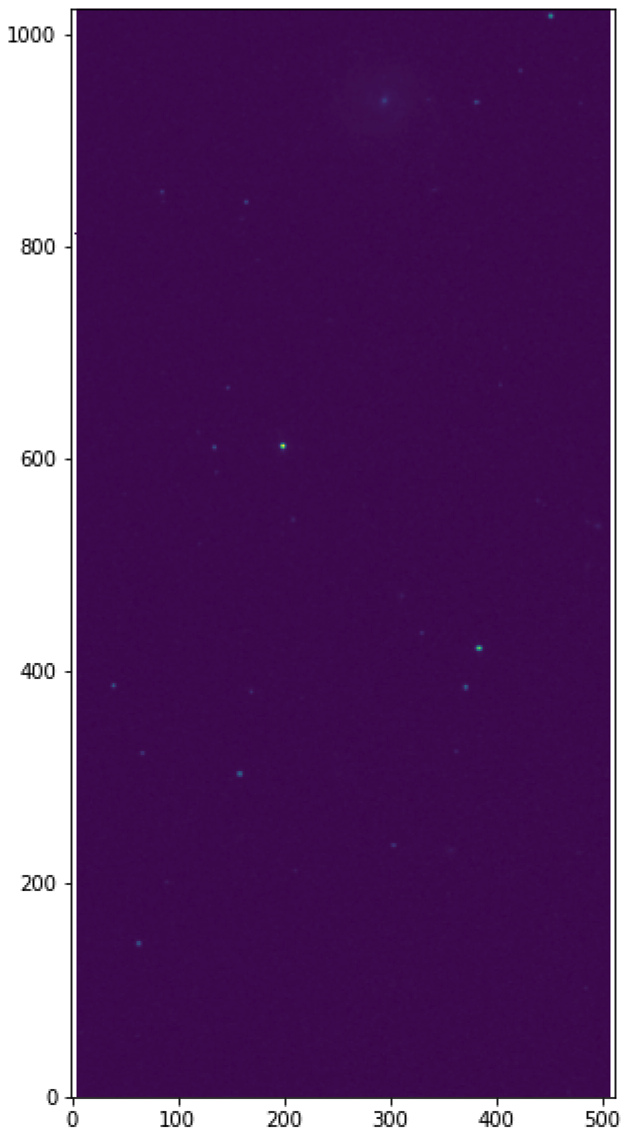
In the analysis of spatially resolved SED of galaxy, it is important to make sure that the multiwavelength images used are all matched to the same spatial resolution (i.e., PSF size) and sampling (i.e., pixel size), so that a given pixel represents the same region on the sky. Such an image processing can be performed using the *piXedfit_images* module. This module is a python scripting module that combines various useful functions in Astropy, photutils, and reproject such that an image processing task for a combination of imaging data can be done automatically.

Before image processing, one need to make sure that the images are all background-free (i.e., background have been subtracted from the images), which are then called as science images. If this is not the case, background subtraction can be performed using `piXedfit.piXedfit_images.subtract_background()` funciton, which will be described further below. After that, one need to construct variance images, which are the square of uncertainty images. For *known images*, this task can be done using various functions in the *piXedfit_images* module. Once science and variance images have been constructed, image processing can be performed using the `piXedfit.piXedfit_images.images_processing` class. The process basically performs PSF matching and spatial resampling and reprojection to the multiwavelength images.

1.7.1 Background subtraction

Suppose we have a 2MASS/J image as shown below (downloaded from 2MASS website), which is not background-free.

```
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt

# open FITS file
hdu = fits.open("aJ_asky_001022s0620186.fits")

# plot the image
plt.figure(figsize=(5,10))
plt.imshow(np.log10(hdu[0].data), origin='lower')
hdu.close()
```



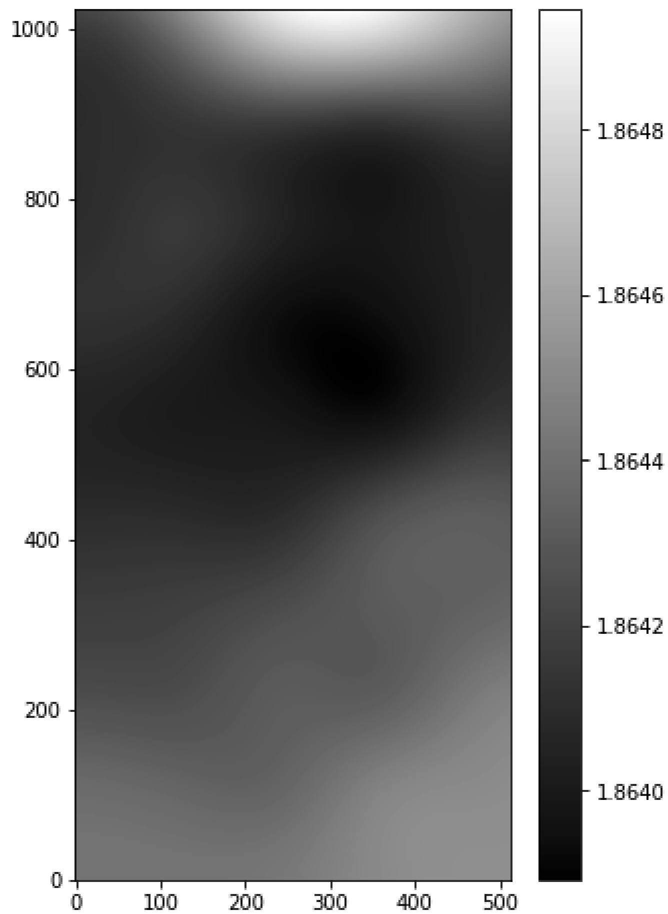
Background subtraction on this image can be performed using *piXedfit.piXedfit_images.subtract_background()* funciton. This function produces background image, RMS image, and background-subtracted science image.

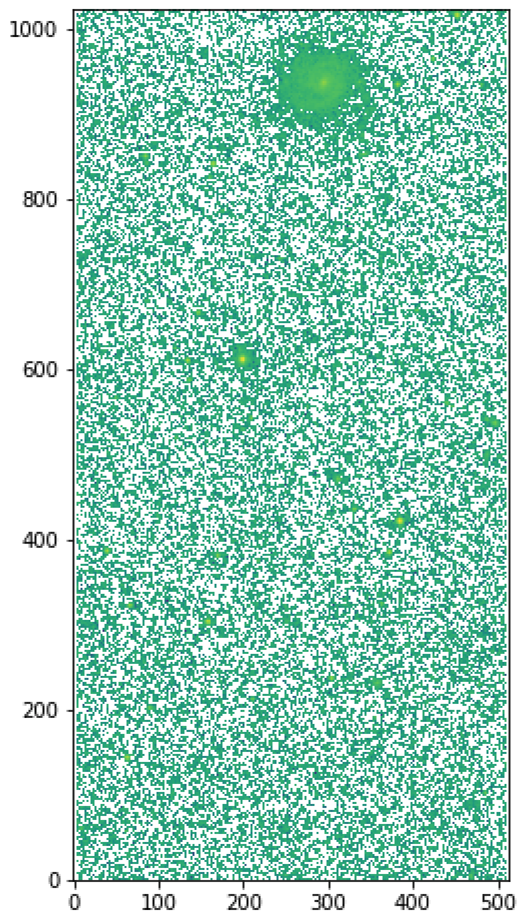
```
from piXedfit.piXedfit_images import subtract_background

fits_image = "aJ_asky_001022s0620186.fits"
subtract_background(fits_image, sigma=3.0, box_size=[100,100], mask_
↪sources=True)
```

`sigma` is the threshold value for the sigma clipping, `box_size` is box size in image gridding, `mask_sources` is a flag stating whether to mask the astroomical sources (i.e., objects) within the image. If set as `True`, source detection and segmentation will be performed using SEP. Please see the API reference for more detail information about this function.

The outputs are: `skybg_aJ_asky_001022s0620186.fits`, `skybgrms_aJ_asky_001022s0620186.fits`, and `skybgsub_aJ_asky_001022s0620186.fits`. Let's plot the background and science images.

```
# background image
hdu = fits.open("skybg_aJ_asky_001022s0620186.fits")
plt.figure(figsize=(5,8))
plt.imshow(np.log10(hdu[0].data), origin='lower', cmap='gray')
plt.colorbar()
hdu.close()

# background-subtracted image
hdu = fits.open("skybgsub_aJ_asky_001022s0620186.fits")
plt.figure(figsize=(5,8))
plt.imshow(np.log10(hdu[0].data), origin='lower')
hdu.close()
```


1000
800
600
400
200
0
0
100
200
300
400
500
1.8648
1.8646
1.8644
1.8642
1.8640



1.7.2 Constructing variance images

For constructing variance (i.e., square of the uncertainty) images, there are various functions provided in **piXedfit**. Depending on the imaging data, one can choose the appropriate function. Available functions are: `var_img_2MASS()`, `var_img_GALEX()`, `var_img_WISE()`, and `var_img_sdss()` for 2MASS, GALEX, WISE, and SDSS imaging data. These functions calculate variance of the pixel values following prescriptions provided in the relevant information or literature associated with the surveys. For other imaging data, one need to construct uncertainty image or weight (i.e., inverse variance) image and then use `var_img_from_unc_img()` or `var_img_from_weight_img()` functions, which are also provided in the *piXedfit_images* module.

In the following, we will demonstrate how to construct variance image from 2MASS and SDSS images. First, we will construct variance image of the 2MASS/J image that we have substracted the background in the previous step.

```
from piXedfit.piXedfit_images import var_img_2MASS

sci_img = "skybgsub_aJ_asky_001022s0620186.fits"
skyrms_img = "skybgrms_aJ_asky_001022s0620186.fits"
var_img_2MASS(sci_img=sci_img, skyrms_img=skyrms_img)
```

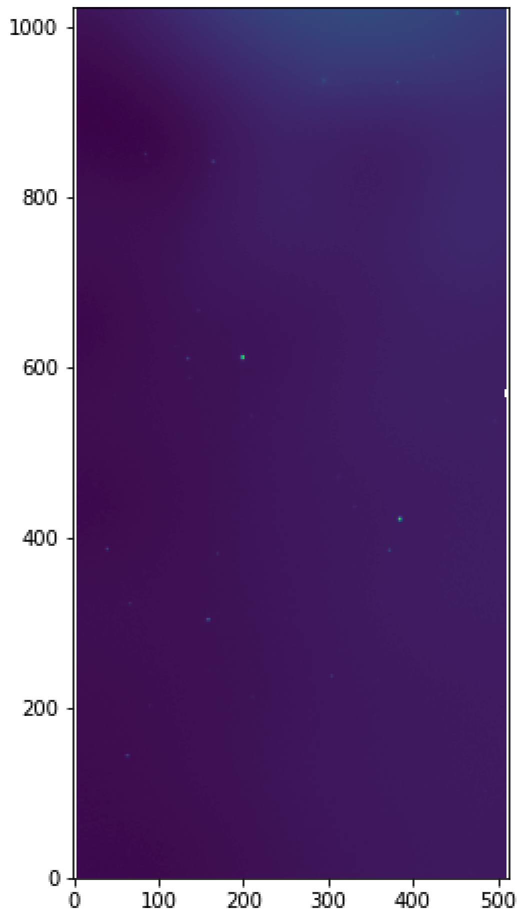
This process will produce `var_skybgsub_aJ_asky_001022s0620186.fits`. Let's plot variance image.

```
hdu = fits.open("var_skybgsub_aJ_asky_001022s0620186.fits")
plt.figure(figsize=(5,8))
```

(continues on next page)

(continued from previous page)

```
plt.imshow(np.log10(hdu[0].data), origin='lower')
hdu.close()
```

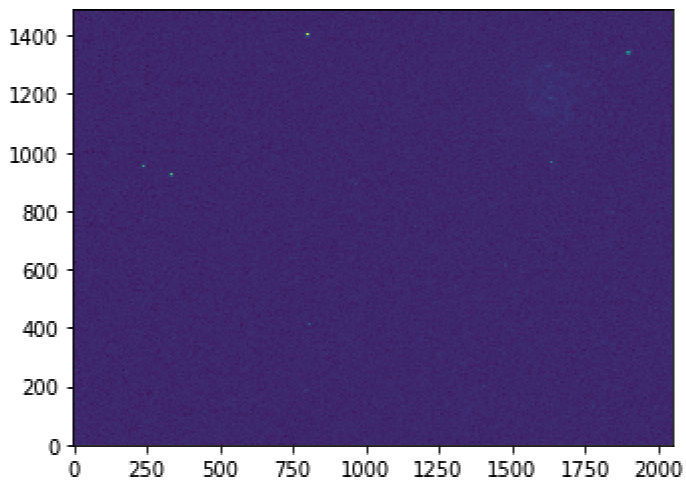


Now, let's try constructing variance image from SDSS image `frame-u-001740-3-0115.fits` (downloaded from the SDSS website).

```
from piXedfit.piXedfit_images import var_img_sdss

fits_image = "frame-u-001740-3-0115.fits"
var_img_sdss(fits_image, filter_name='sdss_u')
```

This will produce `var_frame-u-001740-3-0115.fits`.

```
hdu = fits.open("var_frame-u-001740-3-0115.fits")
plt.figure(figsize=(10,4))
plt.imshow(np.log10(hdu[0].data), origin='lower')
hdu.close()
```



1.7.3 Performing image processing

Next, we will perform image processing. In this example, we will analyze NGC 309 galaxy using 12-band imaging data from GALEX, SDSS, 2MASS, and WISE (W1 and W2). This task can be done using the *piXedfit.piXedfit_images.images_processing* class. In the following, only brief overview of the steps are described. A more complete tutorials can be seen in FUV to NIR images processing or here. The `images_processing` class can also be used for analysis of FUV–FIR data as demonstrated in another tutorial: FUV to FIR images processing.

First, we have to set up the input.

```
# call images_processing
from piXedfit.piXedfit_images import images_processing

# list the filters
filters = ['galex_fuv', 'galex_nuv', 'sdss_u', 'sdss_g', 'sdss_r', 'sdss_i',
        'sdss_z', '2mass_j', '2mass_h', '2mass_k', 'wise_w1', 'wise_w2']

# input science images
sci_img = {}
sci_img['galex_fuv'] = 'GI1_009100_NGC0309-fd-intbgsub.fits'
sci_img['galex_nuv'] = 'GI1_009100_NGC0309-nd-intbgsub.fits'
sci_img['sdss_u'] = 'frame-u-001740-3-0115.fits'
sci_img['sdss_g'] = 'frame-g-001740-3-0115.fits'
sci_img['sdss_r'] = 'frame-r-001740-3-0115.fits'
sci_img['sdss_i'] = 'frame-i-001740-3-0115.fits'
sci_img['sdss_z'] = 'frame-z-001740-3-0115.fits'
sci_img['2mass_j'] = 'skybgsub_aJ_asky_001022s0620186.fits'
sci_img['2mass_h'] = 'skybgsub_aH_asky_001022s0620186.fits'
sci_img['2mass_k'] = 'skybgsub_aK_asky_001022s0620186.fits'
sci_img['wise_w1'] = 'skybgsub_0138m107_ac51-w1-int-3_ra14.177751925_dec-9.
↪913864294_asec1000.000.fits'
sci_img['wise_w2'] = 'skybgsub_0138m107_ac51-w2-int-3_ra14.177751925_dec-9.
↪913864294_asec1000.000.fits'

# input Variance images
var_img = {}
```

(continues on next page)

(continued from previous page)

```
var_img['galex_fuv'] = 'var_GI1_009100_NGC0309-fd-intbgsub.fits'
var_img['galex_nuv'] = 'var_GI1_009100_NGC0309-nd-intbgsub.fits'
var_img['sdss_u'] = 'var_frame-u-001740-3-0115.fits'
var_img['sdss_g'] = 'var_frame-g-001740-3-0115.fits'
var_img['sdss_r'] = 'var_frame-r-001740-3-0115.fits'
var_img['sdss_i'] = 'var_frame-i-001740-3-0115.fits'
var_img['sdss_z'] = 'var_frame-z-001740-3-0115.fits'
var_img['2mass_j'] = 'var_skybgsub_aJ_asky_001022s0620186.fits'
var_img['2mass_h'] = 'var_skybgsub_aH_asky_001022s0620186.fits'
var_img['2mass_k'] = 'var_skybgsub_aK_asky_001022s0620186.fits'
var_img['wise_w1'] = 'var_0138m107_ac51-w1-unc-3_ra14.177751925_dec-9.
↪913864294_asec1000.000.fits'
var_img['wise_w2'] = 'var_0138m107_ac51-w2-unc-3_ra14.177751925_dec-9.
↪913864294_asec1000.000.fits'

# NGC 309 galaxy coordinate
gal_ra = 14.177751925                   # RA
gal_dec = -9.913864294                  # DEC

# redshift of the galaxy
gal_z = 0.0188977

# size of the final stamps will be produced
stamp_size = [131,131]

# initiate the process
img_process = images_processing(filters=filters,sci_img=sci_img,var_img=var_
↪img,gal_ra=gal_ra,
                                        gal_dec=gal_dec, gal_z=gal_z,stamp_
↪size=stamp_size)
```

In the script above, we suply list of filters (see *managing filters*), science images, variance images, the coordinate of the target galaxy, the galaxy's redshift, and the desired size for the final stamp images. One should make sure that the target galaxy is present in the input images, though it is not necessary to trim the input images and make the galaxy to be placed at the center of each image. After the spatial matching, **piXedfit** would automatically locate the galaxy (based on the input coordinate) and crop the region around it when producing the final stamp images.

Image processing can be run using the following command.

```
output_stamps = img_process.reduced_stamps()
```

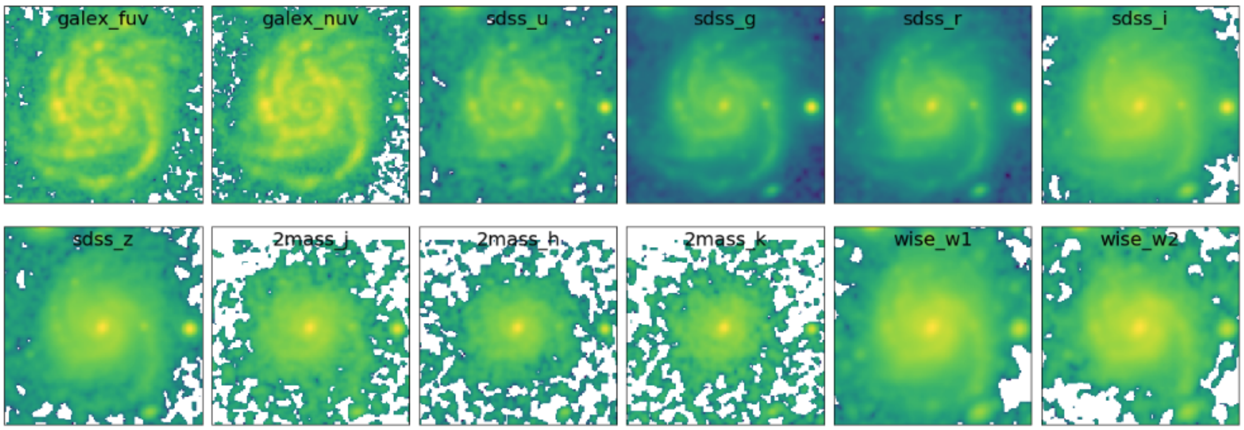
This process produces stamp images that all have the same PSF size and the spatial sampling, in case of our data sets, a PSF FWHM of 6.37" (WISE/W2) and pixel size of 1.5" (GALEX). Now, let's check the stamp images.

```
fig1 = plt.figure(figsize=(20,7))

nbands = len(filters)
for bb in range(0,nbands):
    f1 = fig1.add_subplot(2, 6, bb+1)
    plt.tick_params(left=False,right=False,labelleft=False,labelbottom=False,
↪bottom=False)
    str_temp = "name_img_%s" % filters[bb]
    hdu = fits.open(output_stamps[str_temp])
```

(continues on next page)

(continued from previous page)

```
    plt.imshow(np.log10(hdu[0].data), origin='lower')
    f1.text(0.5, 0.93, filters[bb], horizontalalignment='center',
            verticalalignment='center',transform = f1.transAxes,
            fontsize=20, color='black')
    hdu.close()

plt.subplots_adjust(left=0.05, right=0.95, bottom=0.05, top=0.95, hspace=0.05,␣
↪wspace=0.05)
```



Next, we will define galaxy's region of interest. There are various ways to do this, including the usage of elliptical or circular aperture centered at the galaxy, and more sophesticated way using segmentation maps produced using SEP. In this demo, we will define the galaxy's region through the segmentation process.

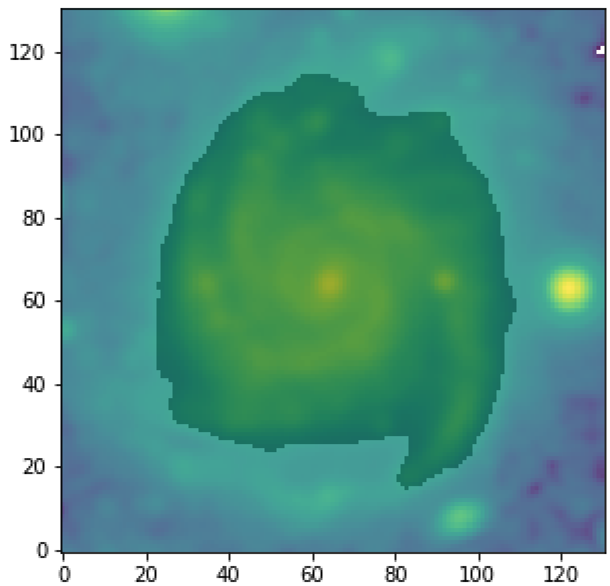
```
segm_maps = img_process.segmentation_sep(output_stamps, thresh=2.8,␣
↪minarea=100,
                                        deblend_nthresh=40, deblend_cont=0.005)
```

This function produces segmentation map on each band, so we get 12 maps. Then, user has a flexibility to choose whether to use single map or merge mutiple maps together for defining the galaxy's region of interest. All this option is possible with the `galaxy_region()` method. Suppose we choose segmentation maps from SDSS i and z bands to be merged, as shown in the following.

```
# select segmentation maps
select_ids = [5, 6]
select_segm_maps = []
for ii in select_ids:
        select_segm_maps.append(segm_maps[ii])

gal_region = img_process.galaxy_region(select_segm_maps)
```

Let's plot the defined region on top of the SDSS/g image.

```
fig1 = plt.figure(figsize=(5,5))
f1 = plt.subplot()
str_temp = "name_img_%s" % filters[3]
hdu = fits.open(output_stamps[str_temp])
plt.imshow(np.log10(hdu[0].data), origin='lower')
plt.imshow(gal_region, origin='lower', cmap='Greys', alpha=0.2)
hdu.close()
```



We are now ready to calculate fluxes (i.e., convert from the pixel values) and flux uncertainties of individual pixels within the galaxy's region of interest. This is can be done using the `flux_map()` method.

```
Gal_EBV = 0.034          # level of attenuation by the foreground Galactic dust
name_out_fits = "fluxmap_ngc309.fits"    # name for the output FITS file
flux_maps = img_process.flux_map(output_stamps, gal_region, Gal_EBV=Gal_EBV,
                                                                    name_out_
↪fits=name_out_fits)
```

`Gal_EBV` is the E(B-V) dust attenuation level due to the foreground Galactic dust. Given the coordinate of the galaxy, this information can be obtained from e.g., NED website. This web application provides attenuation (A_λ) at 5 SDSS bands, which then can be converted into single E(B-V) value using *piXedfit.piXedfit_images.EBV_foreground_dust()* function.

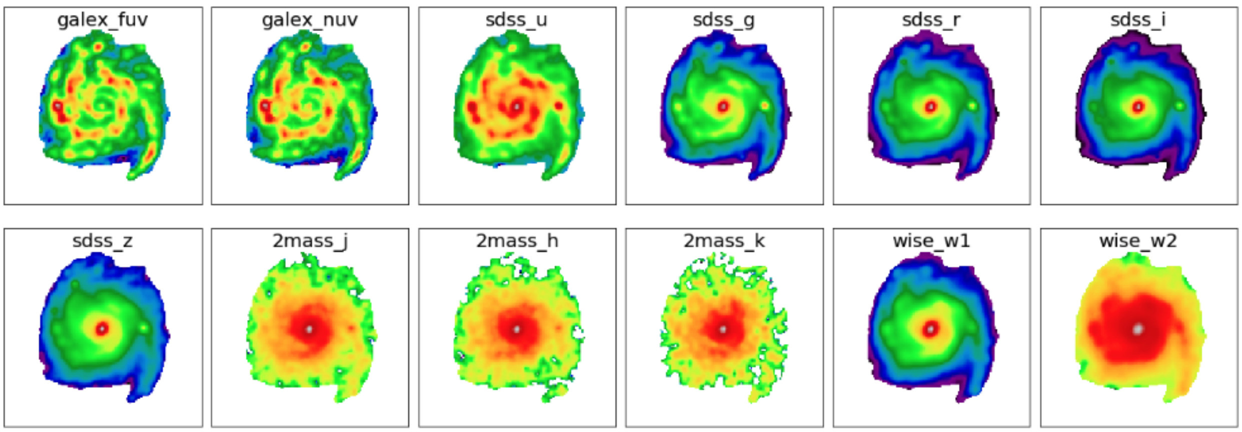
The above process will produce a photometric data cube `fluxmap_ngc309.fits`.

We can check the data cube by plotting maps of the multiband fluxes and the SED on individual pixels. Let's first open the FITS file and extract the information.

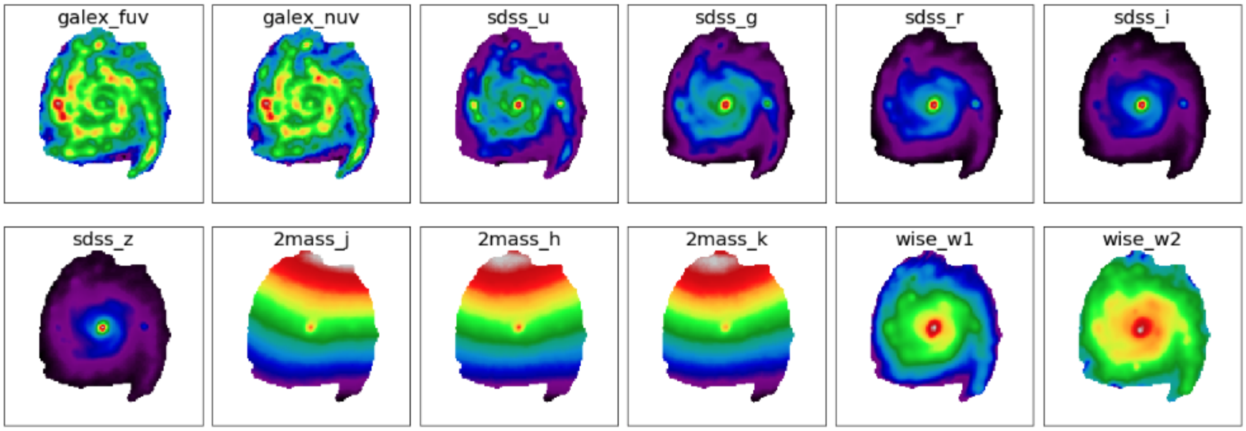
```
# open the FITS file
hdu = fits.open("fluxmap_ngc309.fits")
header = hdu[0].header

# get unit of flux
unit = float(header['unit'])            # in erg/s/cm2/A

# get galaxy's region
gal_region = hdu['GALAXY_REGION'].data
# get maps of fluxes
flux_map = hdu['FLUX'].data*unit
# get maps of flux uncertainties
flux_err_map = hdu['FLUX_ERR'].data*unit
hdu.close()
```

We can then plot maps of the multiband fluxes and flux uncertainties.

```
fig1 = plt.figure(figsize=(20,7))
for bb in range(0,nbands):
        f1 = fig1.add_subplot(2, 6, bb+1)
        plt.tick_params(left=False,right=False,labelleft=False,labelbottom=False,bottom=False)
        plt.imshow(np.log10(flux_map[bb]), origin='lower', cmap='nipy_spectral')
        f1.text(0.5, 0.93, filters[bb], horizontalalignment='center',
                        verticalalignment='center',transform = f1.transAxes,
                        fontsize=20, color='black')

plt.subplots_adjust(left=0.05, right=0.95, bottom=0.05, top=0.95, hspace=0.05, wspace=0.05)
```




```
fig1 = plt.figure(figsize=(20,7))
for bb in range(0,nbands):
        f1 = fig1.add_subplot(2, 6, bb+1)
        plt.tick_params(left=False,right=False,labelleft=False,labelbottom=False,bottom=False)
        plt.imshow(np.log10(flux_err_map[bb]), origin='lower', cmap='nipy_spectral')
        f1.text(0.5, 0.93, filters[bb], horizontalalignment='center',
                        verticalalignment='center',transform = f1.transAxes,
                        fontsize=20, color='black')

plt.subplots_adjust(left=0.05, right=0.95, bottom=0.05, top=0.95, hspace=0.05, wspace=0.05)
```



Next, we will plot SED of some pixels. First, we will transpose the arrays to make it easy for extracting SED of individual pixels given their coordinates.

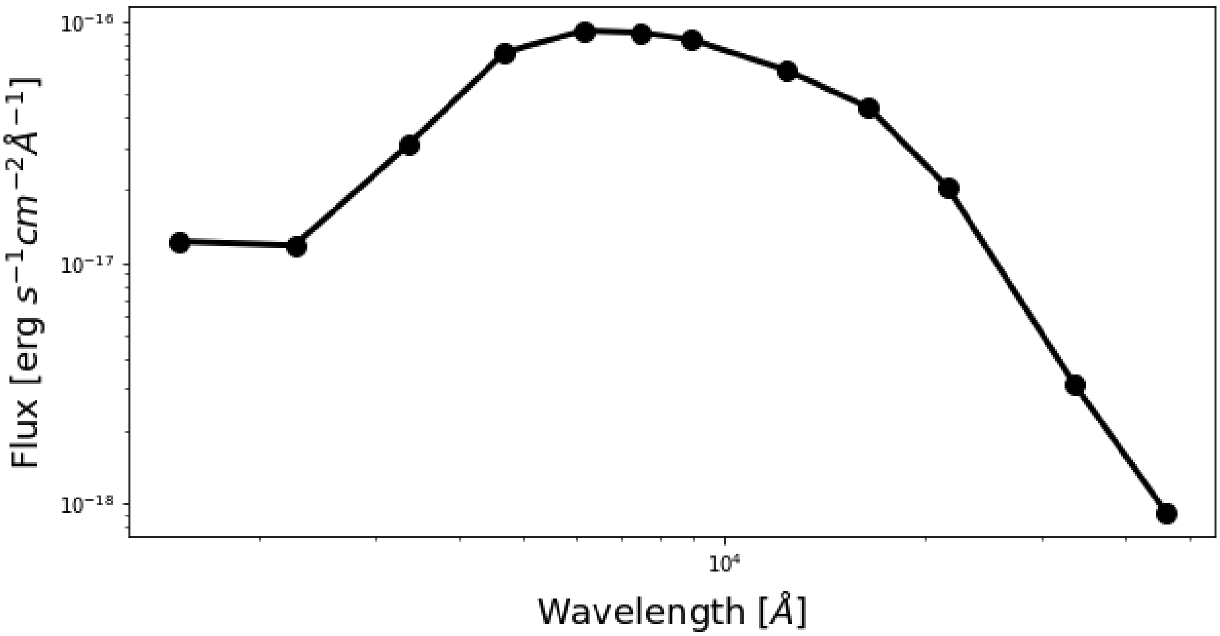
```
# transpose from (band,y,x) to (y,x,band):
pix_SED_flux = np.transpose(flux_map, axes=(1,2,0))
pix_SED_flux_err = np.transpose(flux_err_map, axes=(1,2,0))
```

Before we can plot SED, we need to get central wavelength of the filters. This can be obtained using *piXedfit.utils.filtering.cwave_filters()* function.

```
from piXedfit.utils.filtering import cwave_filters

photo_wave = cwave_filters(filters)
```

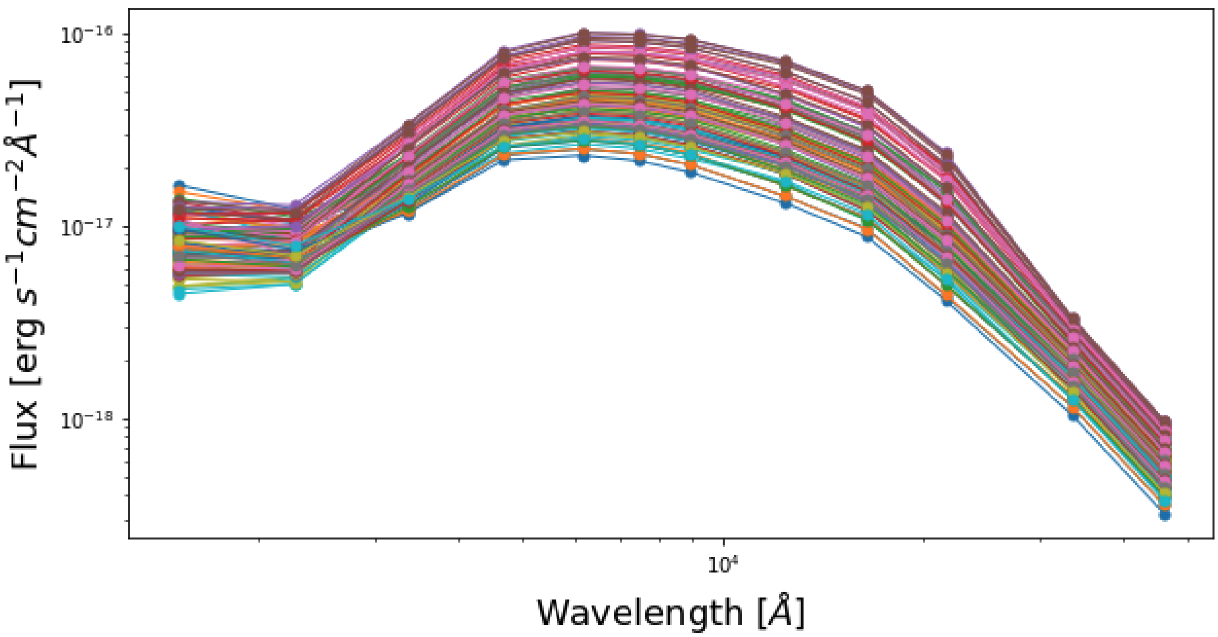
Now we will plot some SEDs. The script below will plot SED of the central pixel.

```
fig1 = plt.figure(figsize=(10,5))
f1 = plt.subplot()
f1.set_yscale('log')
f1.set_xscale('log')
plt.xlabel(r"Wavelength [$\AA$]", fontsize=18)
plt.ylabel(r"Flux [erg $s^{-1}cm^{-2}\AA^{-1}$]", fontsize=18)

# coordinate
pos_y = 65
pos_x = 65

plt.errorbar(photo_wave, pix_SED_flux[pos_y][pos_x], yerr=pix_SED_flux_err[pos_
→y][pos_x]*1e-17,
                 fmt='-o', markersize=10, lw=3, color='black')
plt.show()
```



We will now plot SEDs of pixels within the central 10 x 10.

```
fig1 = plt.figure(figsize=(10,5))
f1 = plt.subplot()

f1.set_yscale('log')
f1.set_xscale('log')
plt.xlabel(r"Wavelength [$\AA$]", fontsize=18)
plt.ylabel(r"Flux [erg $s^{-1}cm^{-2}\AA^{-1}$]", fontsize=18)

for yy in range(60,70):
    for xx in range(60,70):
        pos_y = yy
        pos_x = xx
        plt.errorbar(photo_wave, pix_SED_flux[pos_y][pos_x], yerr=pix_SED_flux_
→err[pos_y][pos_x]*1e-17,
                         fmt='-o', markersize=5, lw=1)

plt.show()
```



1.8 Spatial and Spectral Matching of imaging and IFS data

In the analysis of global (i.e., integrated) SED of galaxies, there have been several attemps at combining spectra (at rest-frame optical) and broadband photometry (over a wider wavelength range) into a so-called spectrophotometric SED. One of the great advantages from this combination is the enhancement of the constraining power in the SED fitting given by the high spectral sampling of the optical spectra and the wider wavelength coverage (e.g., FUV to NIR) of the photometric SED. This could help in breaking the existing degeneracies among parameters (especially among age, metallicity, and dust attenuation) in the SED fitting process.

The current abundance of the imaging and IFS (integral field spectroscopy) data (especially for local galaxies) open up new avenues for the analysis of the spatially resolved spectrophotometric SED of galaxies. In this light, **piXedfit** provides a **new feature** of matching (in both spatially and spectrally) between imaging and IFS data on pixel level. This kind of data would be very useful for various analyses. One of the power of this data, besides enhancing the constraining power as mentioned above, is the wider spatial coverage of the imaging compared to the IFS observation. The imaging observaitons, especially those that are deep and have high spatial resolution, could cover the extended region of galaxies with a sufficient S/N ratio, thus covering most of the optical region of the galaxies. A module in **piXedfit** featuring this task is *piXedfit_spectrophotometric*.

In this brief tutorial, we will combine 12-band imaging data (covering FUV to NIR) and IFS data from the CALIFA survey. The imaging data are the same as we have analyzed in *image processing*, which are consists of imaging data from the GALEX, SDSS, 2MASS, and WISE. A more detailed tutorial can be seen at FUVtoNIR_CALIFA. This feature is also applicable for IFS data from MaNGA survey. A similar tutorial that uses this IFS data is given at FUVtoNIR_MaNGA.

1.8.1 Spatial matching

Overall, there are two steps for combining imaging and IFS data on pixel level. First, spatial matching, which is a matching on the spatial resolution (i.e., PSF size) and sampling (i.e., pixel size). We will do this task here. Second is a matching spectrally, in which we correct for the mismatch between the spectra and photometry along the spectral axis (i.e., wavelength or the 3rd axis). We will perform this task in the next step.

Here we use photometric data cube (`fluxmap_ngc309.fits`) produced in the *image processing* and IFS data from CALIFA (`NGC0309.COMB.rscube.fits.gz`), which is downloaded from the CALIFA website. The spatial matching can be performed using `piXedfit.piXedfit_spectrophotometric.match_imgifs_spatial()`. This process will produce spectrophotometric data cube `specphoto_fluxmap_ngc309.fits`.

```
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt

# call the function
from piXedfit.piXedfit_spectrophotometric import match_imgifs_spatial

photo_fluxmap = "fluxmap_ngc309.fits"                # photometric data cube
ifs_data = "NGC0309.COMB.rscube.fits.gz"             # IFS data cube
ifs_survey = "califa"                                # IFS data source
name_out_fits = "specphoto_fluxmap_ngc309.fits"      # name for the output␣
↪file
match_imgifs_spatial(photo_fluxmap, ifs_data, ifs_survey=ifs_survey,
                        name_out_fits=name_out_fits)
```

Let's check the result. First, let's extract the information from the FITS file.

```
# open the FITS file
cube = fits.open("specphoto_fluxmap_ngc309.fits")

# get header
header = cube[0].header

# get photometry and IFS regions
photo_region = cube['photo_region'].data
spec_region = cube['spec_region'].data

# get unit of flux
unit = float(header['unit'])    # in erg/s/cm2/A

# get maps of photometric fluxes
map_fluxes = cube['photo_flux'].data

# get photometric SEDs of individual pixels
# transpose from (band,y,x) to (y,x,band)
pix_photo_flux = np.transpose(cube['photo_flux'].data, axes=(1,2,0))*unit
pix_photo_flux_err = np.transpose(cube['photo_fluxerr'].data, axes=(1,2,
↪0))*unit

# get spectra of individual pixels
# transpose from (wave,y,x) to (y,x,wave)
pix_spec_flux = np.transpose(cube['spec_flux'].data, axes=(1,2,0))*unit
```

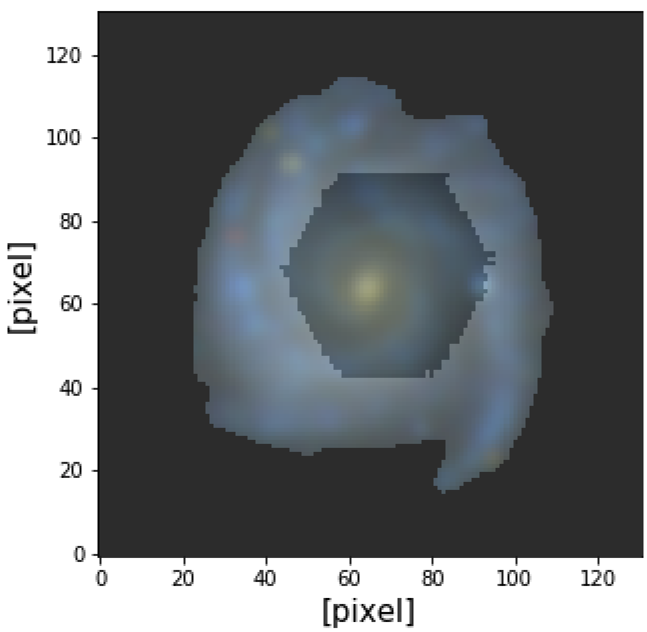
(continues on next page)

(continued from previous page)

```
pix_spec_flux_err = np.transpose(cube['spec_fluxerr'].data, axes=(1,2,0))*unit

# get wavelength of the spectra
spec_wave = cube['wave'].data

cube.close()
```

First, we will plot the coverages of the imaging and IFS data after the matching.

```
from astropy.visualization import make_lupton_rgb

g = map_fluxes[3]*10            # SDSS/g
r = map_fluxes[4]*10            # SDSS/r
i = map_fluxes[5]*10            # SDSS/i

rgb_default = make_lupton_rgb(i, r, g)

fig1 = plt.figure(figsize=(5,5))
f1 = plt.subplot()
plt.xlabel('[pixel]', fontsize=15)
plt.ylabel('[pixel]', fontsize=15)

plt.imshow(rgb_default, origin='lower', alpha=1.0)
plt.imshow(spec_region, origin='lower', cmap='Greys', alpha=0.2)
```



Now, we will check SED of some pixels. Before we can plot the SEDs, we need to get the list of filters and their central wavelength.

```
# get filters from the header
nbands = int(header['nfilters'])
filters = []
for bb in range(0,nbands):
```

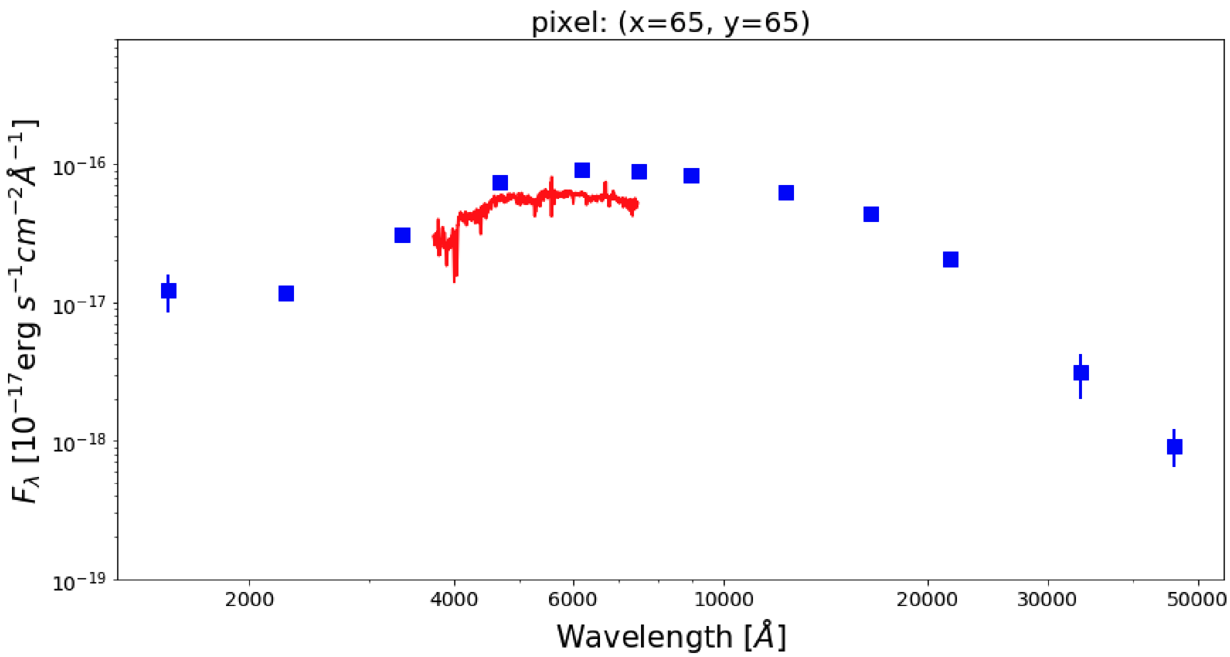
(continues on next page)

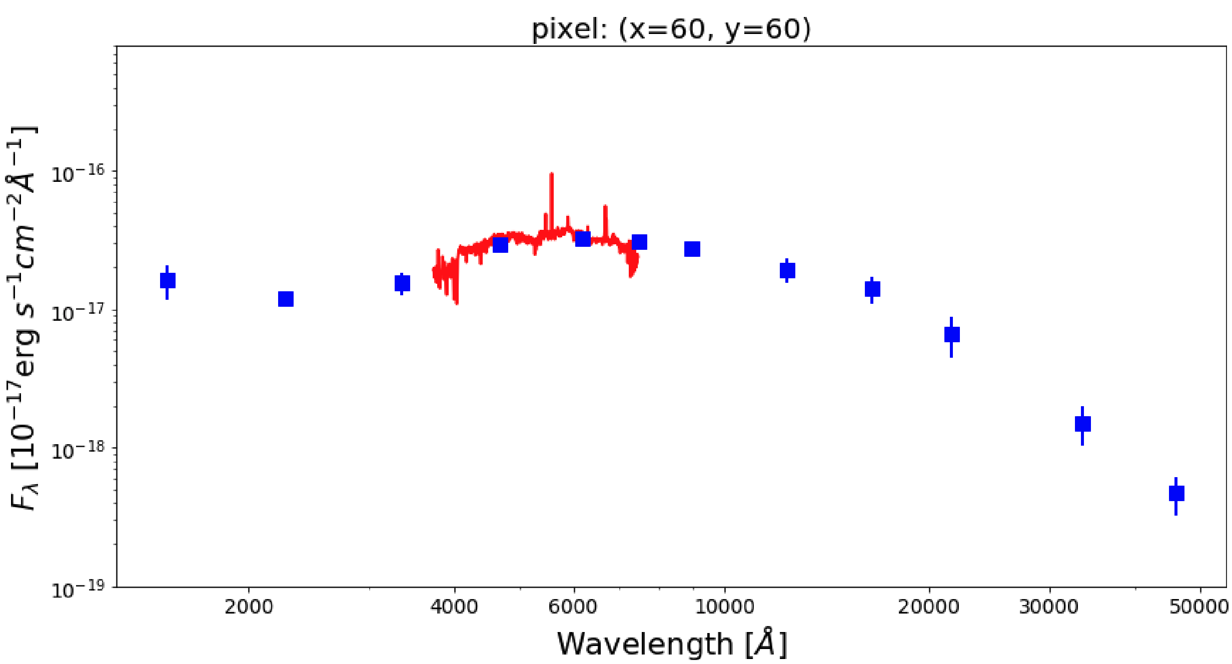
(continued from previous page)

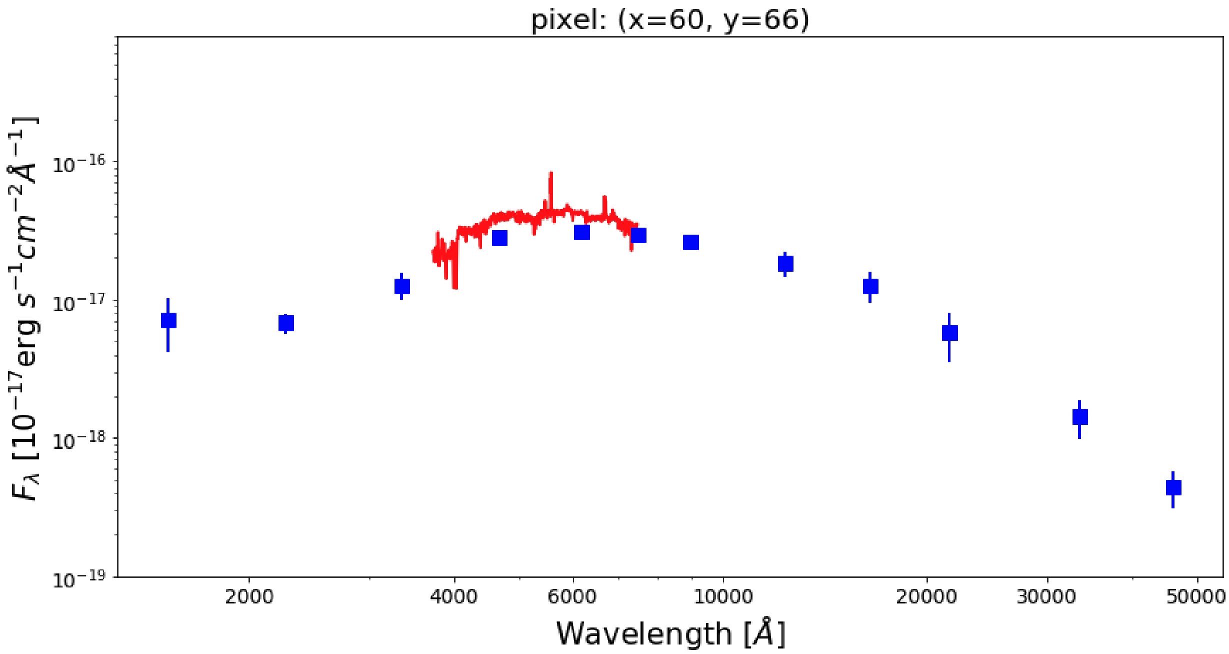
```
        str_temp = 'fil%d' % bb
        filters.append(header[str_temp])

# get central wavelength of filters
from piXedfit.utils.filtering import cwave_filters
photo_wave = cwave_filters(filters)
```

Below we plot SEDs of three example pixels.

```
from matplotlib.ticker import ScalarFormatter

pix_x = [65, 60, 60]
pix_y = [65, 60, 66]

nwaves = len(spec_wave)
for ii in range(0,3):
        xx, yy = pix_x[ii], pix_y[ii]

        photo_SED = pix_photo_flux[yy][xx]
        photo_SED_err = pix_photo_flux_err[yy][xx]
        spec_SED = pix_spec_flux[yy][xx]

        fig1 = plt.figure(figsize=(14,7))
        f1 = plt.subplot()
        plt.title("pixel: (x=%d, y=%d)" % (xx,yy), fontsize=20)
        f1.set_yscale('log')
        f1.set_xscale('log')
        plt.setp(f1.get_yticklabels(), fontsize=14)
        plt.setp(f1.get_xticklabels(), fontsize=14)
        plt.xlabel(r'Wavelength $[\AA]$', fontsize=21)
        plt.ylabel(r'$F_{\lambda}$ [$10^{-17}$erg $s^{-1}cm^{-2}\AA^{-1}$]',␣
↪fontsize=21)
        xticks = [2000,4000,6000,10000,20000,30000,50000]
        plt.xticks(xticks)
        plt.ylim(1.0e-19,8e-16)
        for axis in [f1.xaxis]:
                axis.set_major_formatter(ScalarFormatter())

        # Optional: cut the spectra around the edge to exclude those commonly␣
↪uncertain fluxes
        plt.plot(spec_wave[20:nwaves-20], spec_SED[20:nwaves-20], lw=2.0,␣
↪color='red')
        plt.errorbar(photo_wave, photo_SED, yerr=photo_SED_err, markersize=10,
                                        color='blue', fmt='s', lw=2)
```


pixel: (x=65, y=65)
F_λ [10^{-17}erg $s^{-1}cm^{-2}Å^{-1}$]
10^{-16}
10^{-17}
10^{-18}
10^{-19}
2000
4000
6000
10000
20000
30000
50000
Wavelength [Å]


pixel: (x=60, y=60)
F_λ [10^{-17}erg $s^{-1}cm^{-2}Å^{-1}$]
10^{-16}
10^{-17}
10^{-18}
10^{-19}
2000
4000
6000
10000
20000
30000
50000
Wavelength [Å]



1.8.2 Spectral matching

As we see in the figures above, there is a missmatch between the spectra and the photometric SEDs. This discrepancy seems to be complex as it shows variations from galaxy to galaxy and the wavelength-variations nature of the discrepancy. Next, we will correct this discrepancy using *piXedfit.piXedfit_spectrophotometric.match_imgifs_spectral()* function. Basically, this function will perform simple SED fitting to the photometric SED of individual pixels to find a best-fit model spectrum for each pixel. A wavelength-dependent ratio (between the best-fit model spectrum and the IFS spectrum) is then fit with a third-order Legendre polynomial function to get smooth correction factor, which is then applied to correct the IFS spectra.

This calculation is rather heavy and time-consuming, depending on the number of pixels (that have spec+photo SEDs) and the number of cores that are used (nproc) in the calculation. It's recommended to run this calculation separately (from this jupyter notebook) on a multicores computer (or cluster). The computational time can be shorten by inncreasing the number of cores.

```
from piXedfit.piXedfit_spectrophotometric import match_imgifs_spectral

# spectrophotometric data cube produced in the previous step
specphoto_file = "specphoto_fluxmap_ngc309.fits"
nproc = 20                                  # number of cores to be used␣
↪for calculation
name_out_fits = "corr_%s" % specphoto_file          # name of output file

match_imgifs_spectral(specphoto_file, nproc=nproc, name_out_fits=name_out_fits)
```

This process will produce spectrophotometric data cube `corr_specphoto_fluxmap_ngc309.fits`. In the following, we will check this data cube. First, we open the FITS file and extract the data.

```
# open the FITS file
cube = fits.open("corr_specphoto_fluxmap_ngc309.fits")

header = cube[0].header
```

(continues on next page)

(continued from previous page)

```
wave = cube['wave'].data
photo_flux = cube['PHOTO_FLUX'].data
photo_flux_err = cube['PHOTO_FLUXERR'].data
spec_flux = cube['SPEC_FLUX'].data
spec_flux_err = cube['SPEC_FLUXERR'].data
# correction factors
corr_factor = cube['corr_factor'].data

# unit of flux
unit = header['unit']   # in erg/s/cm2/A
cube.close()
```

Get SEDs of individual pixels.

```
# get photometric SEDs of pixels:
#transpose (band,y,x) => (y,x,band):
pix_photo_SED = np.transpose(photo_flux, axes=(1, 2, 0))*unit
pix_photo_SED_err = np.transpose(photo_flux_err, axes=(1, 2, 0))*unit

# get spectra:
#transpose (wavelength,y,x) => (y,x,wavelength):
pix_spec_SED = np.transpose(spec_flux, axes=(1, 2, 0))*unit
pix_spec_SED_err = np.transpose(spec_flux_err, axes=(1, 2, 0))*unit

pix_corr_factor = np.transpose(corr_factor, axes=(1, 2, 0))
```

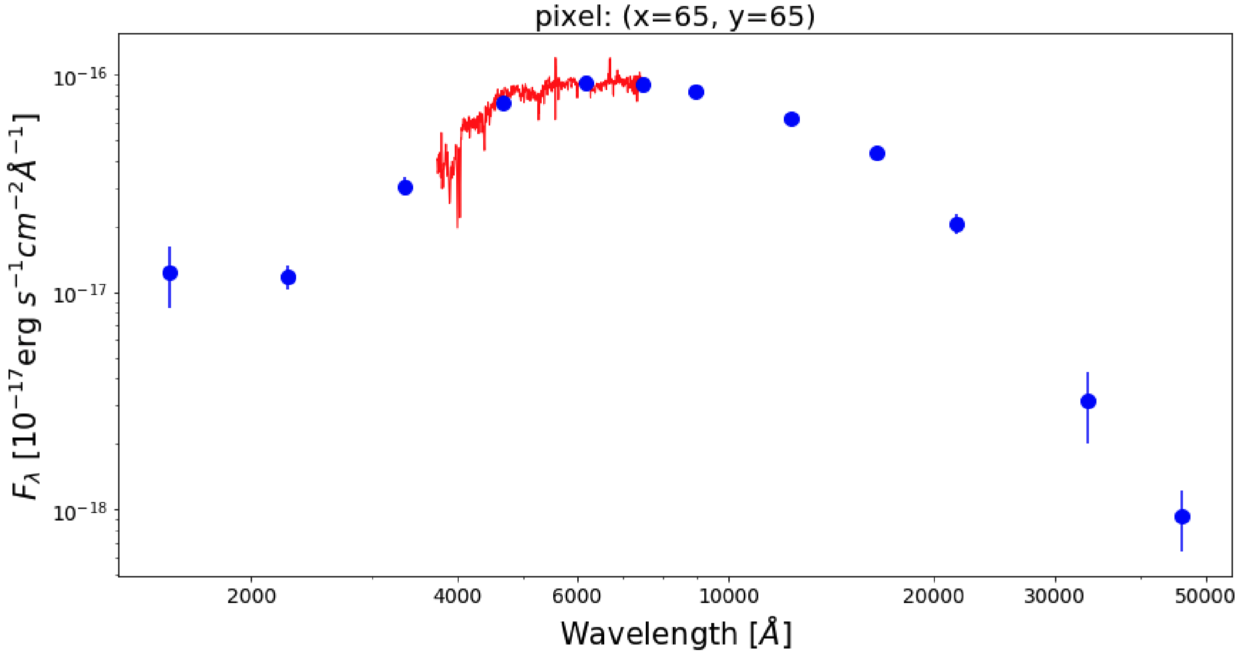
Make plot of the SEDs of the same three pixels that we have shown before, but now after correction has been applied.

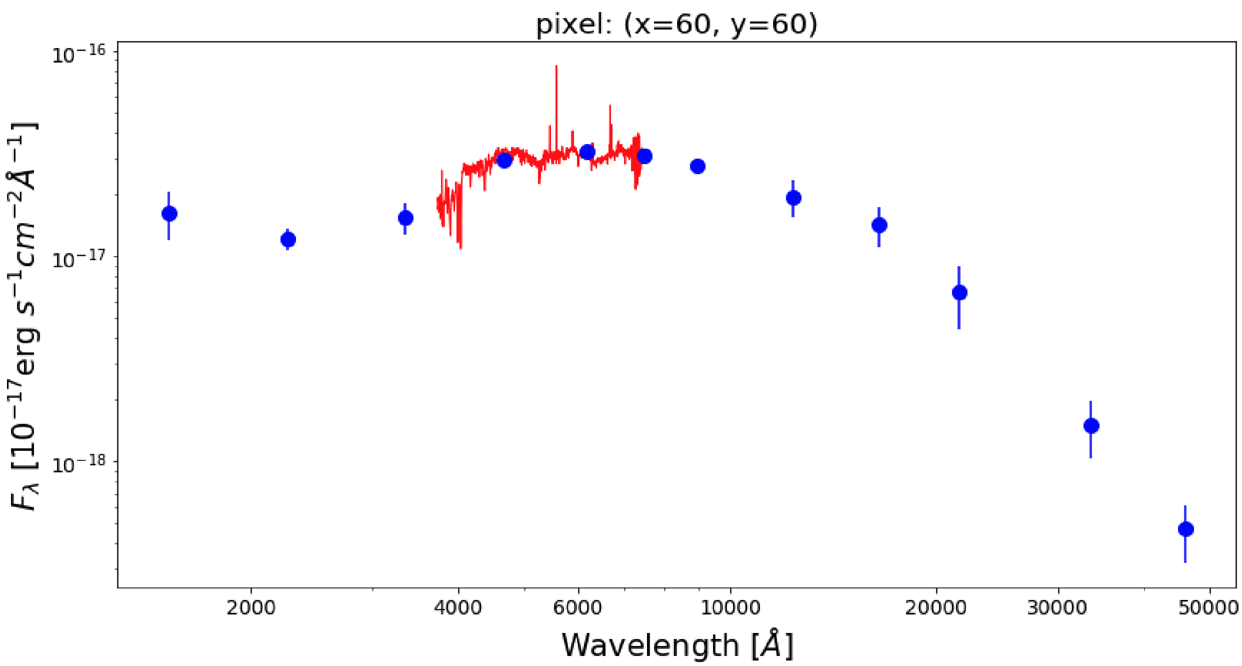
```
nwaves = len(wave)
for ii in range(0,3):
        xx, yy = pix_x[ii], pix_y[ii]

        fig1 = plt.figure(figsize=(14,7))
        f1 = plt.subplot()
        plt.title("pixel: (x=%d, y=%d)" % (xx,yy), fontsize=20)
        f1.set_yscale('log')
        f1.set_xscale('log')
        plt.setp(f1.get_yticklabels(), fontsize=14)
        plt.setp(f1.get_xticklabels(), fontsize=14)
        plt.xlabel(r'Wavelength $[\AA]$', fontsize=21)
        plt.ylabel(r'$F_{\lambda}$ [$10^{-17}$erg $s^{-1}cm^{-2}\AA^{-1}$]',␣
↪fontsize=21)
        xticks = [2000,4000,6000,10000,20000,30000,50000]
        plt.xticks(xticks)
        plt.ylim(1.0e-19,8e-16)
        for axis in [f1.xaxis]:
                axis.set_major_formatter(ScalarFormatter())

        # Optional: cut the spectra around the edge to exclude those commonly␣
↪uncertain fluxes
        plt.plot(wave[20:nwaves-20], pix_spec_SED[yy][xx][20:nwaves-20], lw=2.
↪0, color='red')
```

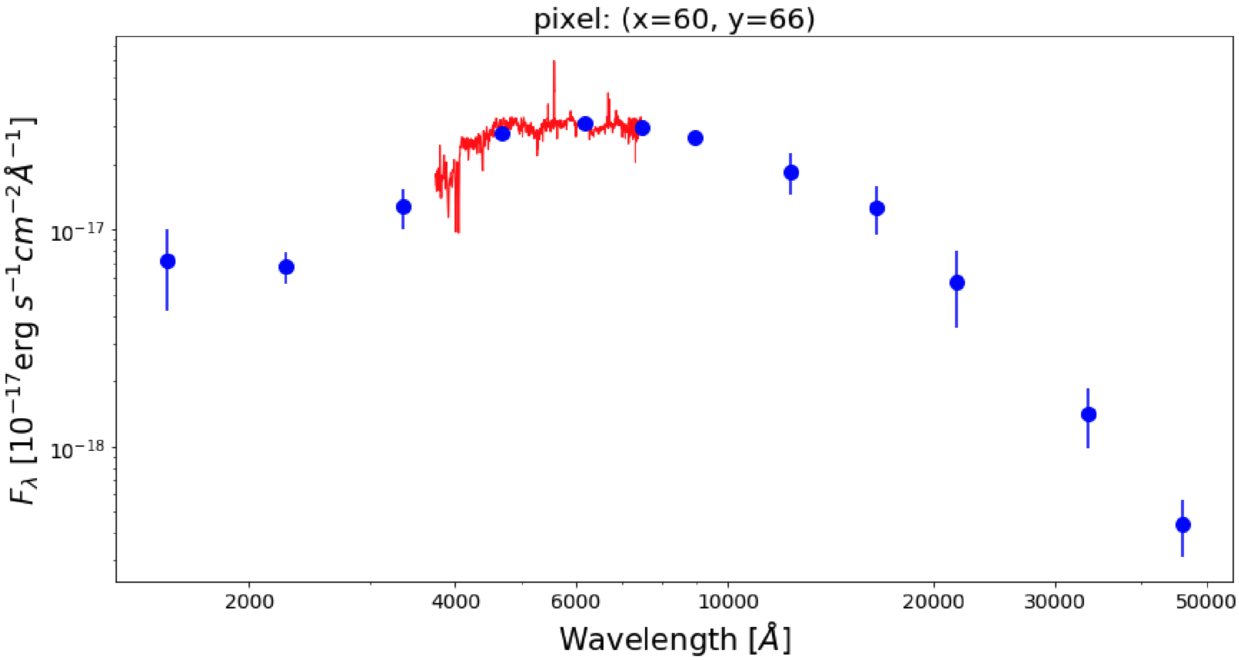
(continues on next page)

(continued from previous page)

```
        plt.errorbar(photo_wave, pix_photo_SED[yy][xx], yerr=pix_photo_SED_
↪err[yy][xx], markersize=10,
                                color='blue', fmt='s', lw=2)
```

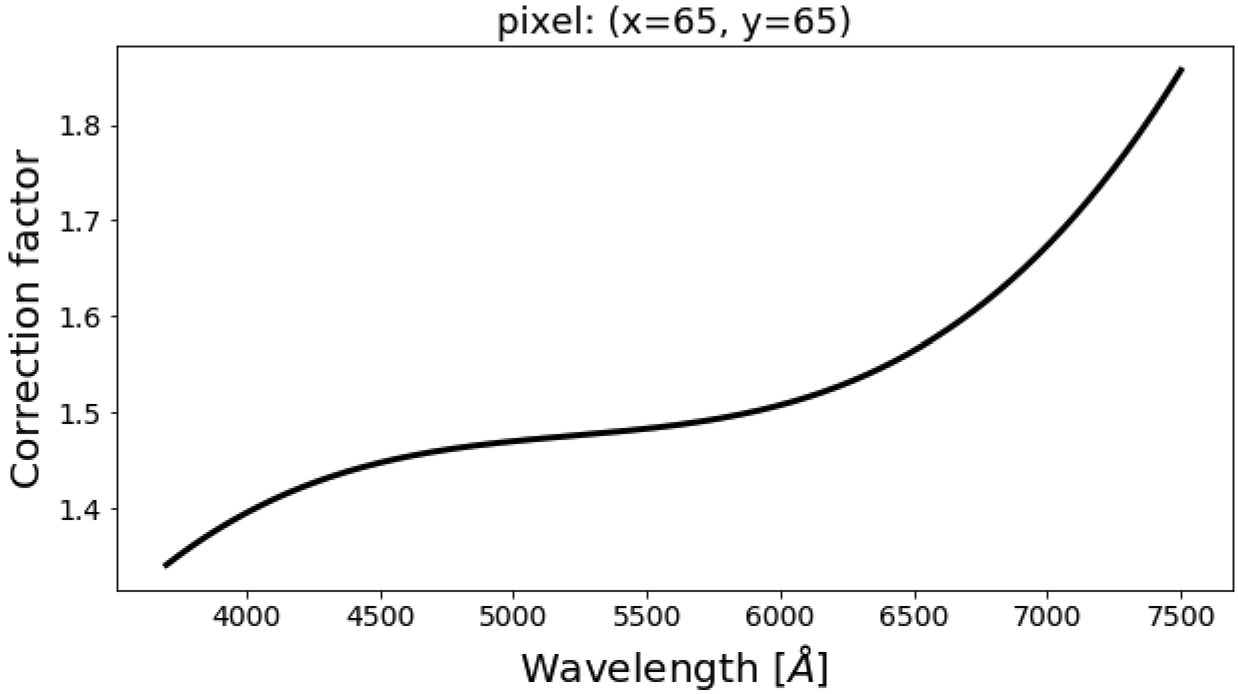


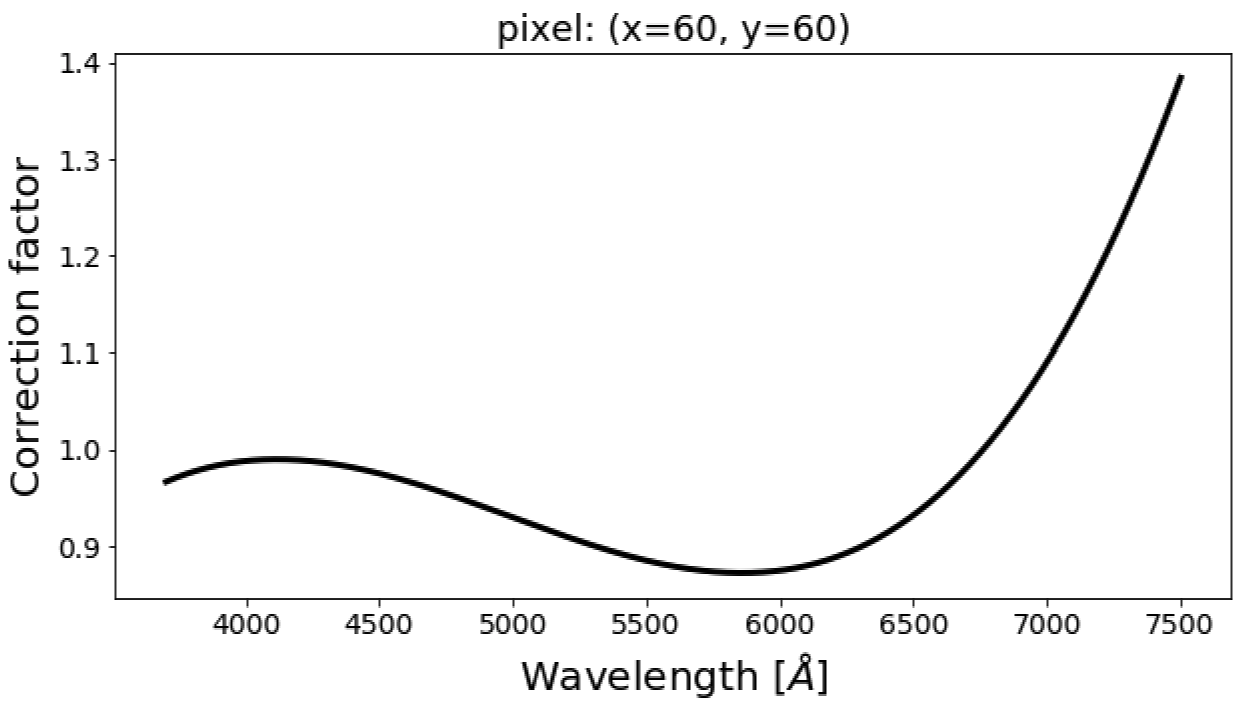


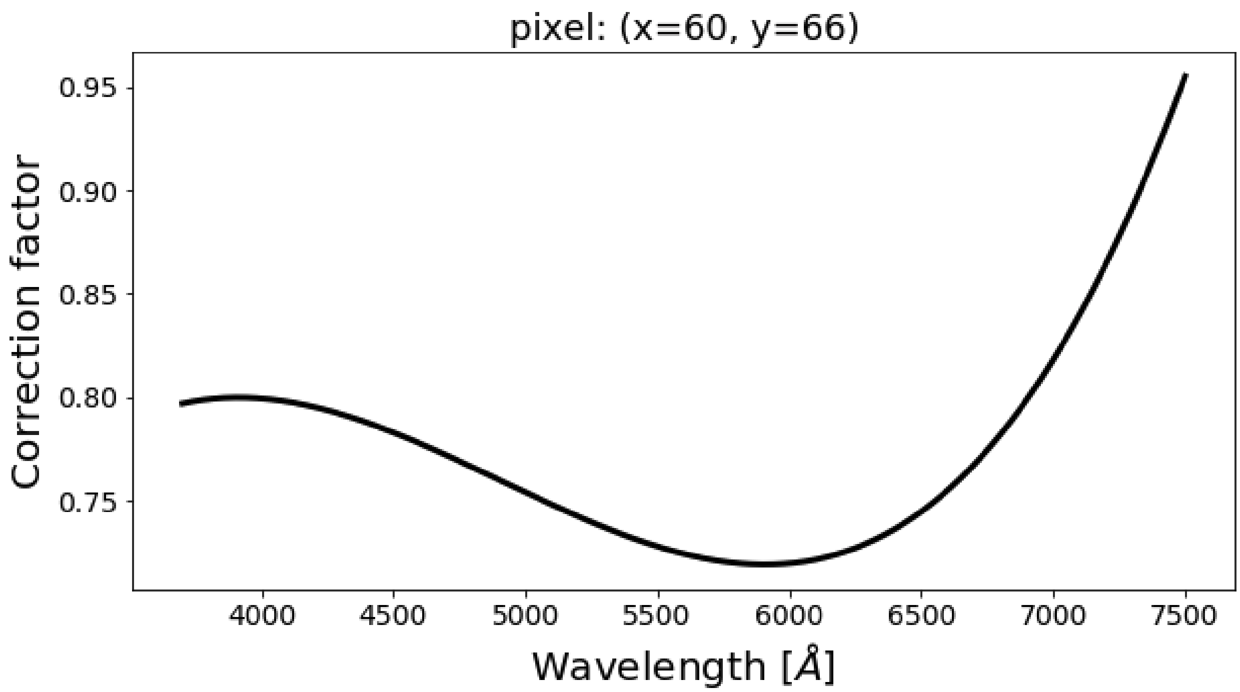


We can see the correction factors that have been applied to the spectra with the following script.

```
for ii in range(0,3):
xx, yy = pix_x[ii], pix_y[ii]

fig1 = plt.figure(figsize=(14,7))
f1 = plt.subplot()
plt.title("pixel: (x=%d, y=%d)" % (xx,yy), fontsize=20)
plt.setp(f1.get_yticklabels(), fontsize=14)
plt.setp(f1.get_xticklabels(), fontsize=14)
plt.xlabel(r'Wavelength $[\AA]$', fontsize=21)
plt.ylabel(r'Correction factor', fontsize=21)

plt.plot(wave, pix_corr_factor[yy][xx], lw=3.0, color='black')
```


pixel: (x=65, y=65)
Correction factor
1.8
1.7
1.6
1.5
1.4
4000
4500
5000
5500
6000
6500
7000
7500
Wavelength [Å]


pixel: (x=60, y=60)
Correction factor
1.4
1.3
1.2
1.1
1.0
0.9
4000
4500
5000
5500
6000
6500
7000
7500
Wavelength [Å]



1.9 Pixel binning

In most cases, fluxes measured in individual pixels have a low S/N ratio. It is also common to find pixels with missing or negative fluxes. In order to gain an accurate inference of the parameters in the SED fitting, one typically needs and observed SED with sufficient S/N. For this reason, we bin SEDs of pixels before conducting further analysis, such as SED fitting. Previous studies have developed pixel binning schemes to deal with the low S/N on spatially resolved scales. However, the previous binning scheme mainly consider two factors only, which are proximity of pixels and a target S/N on a particular band that is goind to be achieved by binning the pixels.

piXedfit introduces a new pixel binning scheme that incorporates a new important constraint, namely the similarity in the SED shape among pixels. This new criterion is important, especially for the spatially resolved SED fitting analyses, because it is expected to preserve any important information in the SED at the pixel level. While pixel binning is done to achieve a certain S/N threshold, at the cost of degrading the spatial resolution, we can still preserve important information in the SED at the pixel level with this binning scheme. In the previous pixel binning schemes that do not consider the similarity of the SED shape, it is possible that neighboring pixels that have different SED shapes (likely having different properties) are binned together. This could smooth out the spatial variation of the stellar population properties.

In practice, the SED shape similarity among pixels is evaluated using the chi-square statistics. User can set a maximum value of chi-square (or a reduced chi-square, `redc_chi2_limit`) below which a pair of SEDs are considered to be similar in shape. Besides the similarity in SED shape, the binning scheme in **piXedfit** also considers: (a) proximity (i.e., only neighboring pixels are binned together), (b) minimum diameter of a bin (`Dmin_bin`), which can be thought of as the FWHM of the PSF (although the user is free to define the diameter), and (c) S/N threshold in each band (`SNR`). The last criterion is also a new feature which allow user to get sufficient S/N across the filters, not limited to a particular filter only.

The pixel binning task in **piXedfit** is done by *piXedfit_bin* module. This module provides functions for performing pixel binning on a 3D data cube (either photometric or spectrophotometric data cube) and imaging data (in single band or multiband). Below, we will demonstrate how to use these functions.

1.9.1 Pixel binning on 3D data cube

In this example, we will perform pixel binning to the spectrophotometric data cube (corr_specphoto_fluxmap_ngc309.fits) that we have produced in the previous step (i.e., *spatial and spectral matching of imaging and IFS data*). This task can be done using *piXedfit.piXedfit_bin.pixel_binning()* function. This function can handle both the photometric and spectrophotometric data cubes. When the input is spectrophotometric data cube, this function will perform pixel binning to the photometric data first and then spectra of pixels are binned following the resulted binning map, so only spatial bins that fall within the IFU region will have both the photometric and spectroscopic SEDs.

Below is an example of the lines of code for running a pixel binning. Here we want to achieve minimum S/N of 10 in 7 filters of GALEX and SDSS bands, while minimum S/N=0 for the other 5 bands. For evaluating the SED shape similarity, here we assume a maximum reduced chi-square of 4. We also set a minimum diameter of 4 pixels for the spatial bin, considering the pixel size of the data cube of 1.5" and the PSF FWHM size of 6.37" (~4 pixels).

```
import numpy as np
from astropy.io import fits
import matplotlib.pyplot as plt

# call pixel binning function
from piXedfit.piXedfit_bin import pixel_binning

fits_fluxmap = "corr_specphoto_fluxmap_ngc309.fits"
ref_band = 4                    # SDSS/r as the ref. band in brightness␣
↪comparison among pixels
Dmin_bin = 4                    # in pixel
redc_chi2_limit = 4.0

# Set S/N thresholds.
nbands = 12             # number of filters in our data
SNR = np.zeros(nbands)
for ii in range(0,7):
        SNR[ii] = 10.0

name_out_fits = "pixbin_%s" % fits_fluxmap
pixel_binning(fits_fluxmap, ref_band=ref_band, Dmin_bin=Dmin_bin, SNR=SNR,
                redc_chi2_limit=redc_chi2_limit, name_out_fits=name_out_fits)
```

The output of this process is pixbin_corr_specphoto_fluxmap_ngc309.fits. Now, we will extract information from the FITS file and see the map of spatial bins and the SEDs of individual bins.

```
hdu = fits.open("pixbin_corr_specphoto_fluxmap_ngc309.fits")
header = hdu[0].header

# get number bins that have photometric and spectrophotometric data
nbins_photo = int(header['nbinsph'])
nbins_spec = int(header['nbinssp'])

# get set of filters
nbands = int(header['nfilters'])
filters = []
for bb in range(0,nbands):
        str_temp = 'fil%d' % bb
        filters.append(header[str_temp])
```

(continues on next page)

(continued from previous page)

```
# get central wavelength of filters
from piXedfit.utils.filtering import cwave_filters
photo_wave = cwave_filters(filters)

# spatial bin maps
binmap_photo = hdu['photo_bin_map'].data
binmap_spec = hdu['spec_bin_map'].data

# unit of flux
unit = float(header['unit'])                    # in erg/s/cm2/A

# wavelength of the spectra
spec_wave = hdu['spec_wave'].data
nwaves = len(spec_wave)

# allocate arrays for photometric and spectrophotometric SEDs of spatial bins
bin_photo_flux = np.zeros((nbins_photo,nbands))
bin_photo_flux_err = np.zeros((nbins_photo,nbands))

bin_spec_flux = np.zeros((nbins_photo,nwaves))
bin_spec_flux_err = np.zeros((nbins_photo,nwaves))

for bb in range(0,nbins_photo):
        bin_id = bb + 1

        rows, cols = np.where(binmap_photo==bin_id)
        bin_photo_flux[bb] = hdu['bin_photo_flux'].data[:,rows[0],cols[0]]*unit
        bin_photo_flux_err[bb] = hdu['bin_photo_fluxerr'].data[:,rows[0],
↪cols[0]]*unit

        rows, cols = np.where(binmap_spec==bin_id)
        if len(rows)>0:
                bin_spec_flux[bb] = hdu['bin_spec_flux'].data[:,rows[0],
↪cols[0]]*unit
                bin_spec_flux_err[bb] = hdu['bin_spec_fluxerr'].data[:,rows[0],
↪cols[0]]*unit

hdu.close()
```

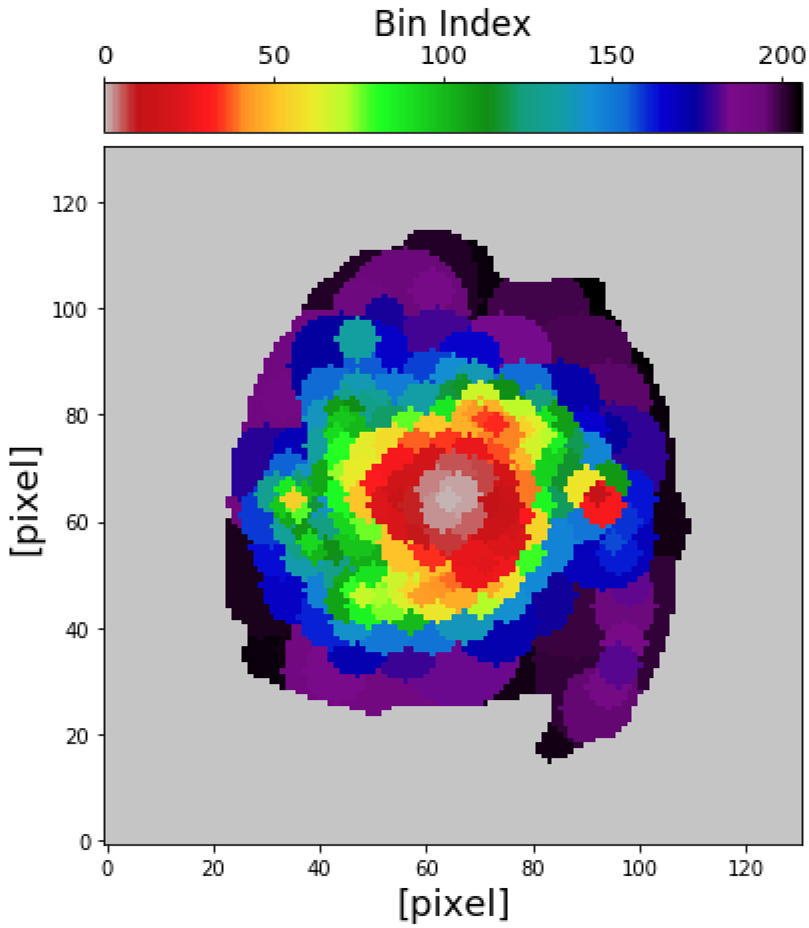
Then we can plot the map of spatial bins in the following way. First, let's plot the map of bins that have photometric data only.

```
from mpl_toolkits.axes_grid1 import make_axes_locatable

fig1 = plt.figure(figsize=(7,7))
f1 = plt.subplot()
plt.xlabel("[pixel]", fontsize=18)
plt.ylabel("[pixel]", fontsize=18)

im = plt.imshow(binmap_photo, origin='lower', cmap='nipy_spectral_r', vmin=0,␣
↪vmax=nbins_photo)
```

(continues on next page)

(continued from previous page)

```
divider = make_axes_locatable(f1)
cax2 = divider.append_axes("top", size="7%", pad="2%")
cb = fig1.colorbar(im, cax=cax2, orientation="horizontal")
cax2.xaxis.set_ticks_position("top")
cax2.xaxis.set_label_position("top")
cb.ax.tick_params(labelsize=13)
cb.set_label('Bin Index', fontsize=17)
```



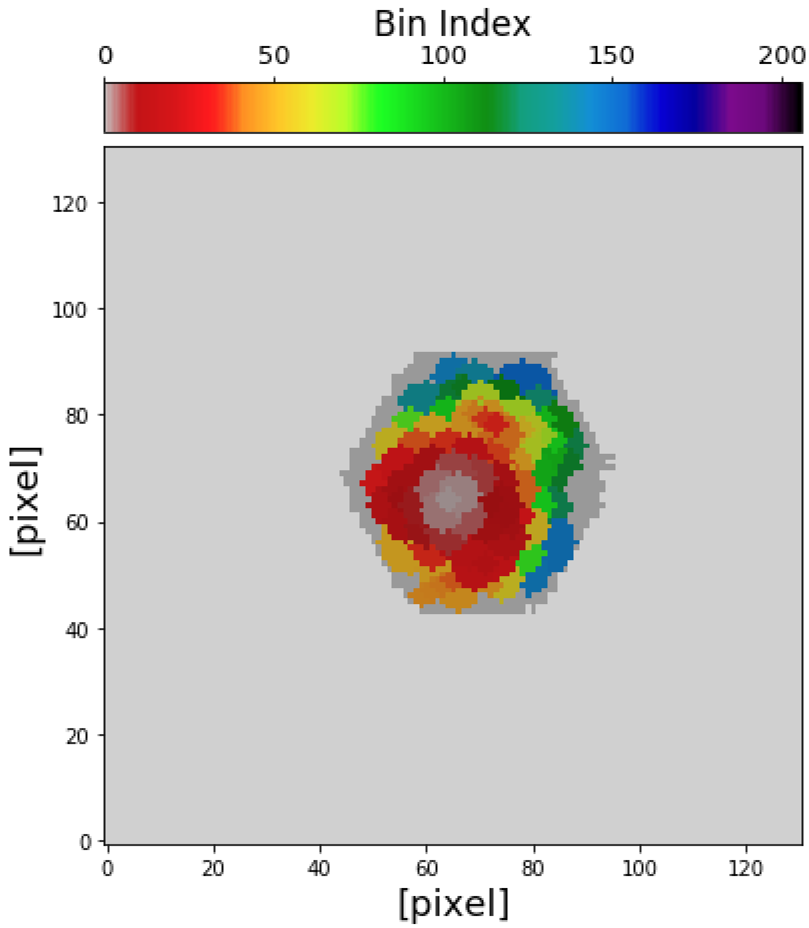
Then the map of spatial bins that have spectrophotometric data can be plot in the following way.

```
# get spec region
hdu = fits.open("corr_specphoto_fluxmap_ngc309.fits")
spec_region = hdu['spec_region'].data
hdu.close()

# plot spatial bin map
fig1 = plt.figure(figsize=(7,7))
f1 = plt.subplot()
plt.xlabel("[pixel]", fontsize=18)
plt.ylabel("[pixel]", fontsize=18)
```

(continues on next page)

(continued from previous page)

```
im = plt.imshow(binmap_spec, origin='lower', cmap='nipy_spectral_r', vmin=0,␣
↪vmax=nbins_photo)
plt.imshow(spec_region, origin='lower', cmap='Greys', alpha=0.2)

divider = make_axes_locatable(f1)
cax2 = divider.append_axes("top", size="7%", pad="2%")
cb = fig1.colorbar(im, cax=cax2, orientation="horizontal")
cax2.xaxis.set_ticks_position("top")
cax2.xaxis.set_label_position("top")
cb.ax.tick_params(labelsize=13)
cb.set_label('Bin Index', fontsize=17)
```



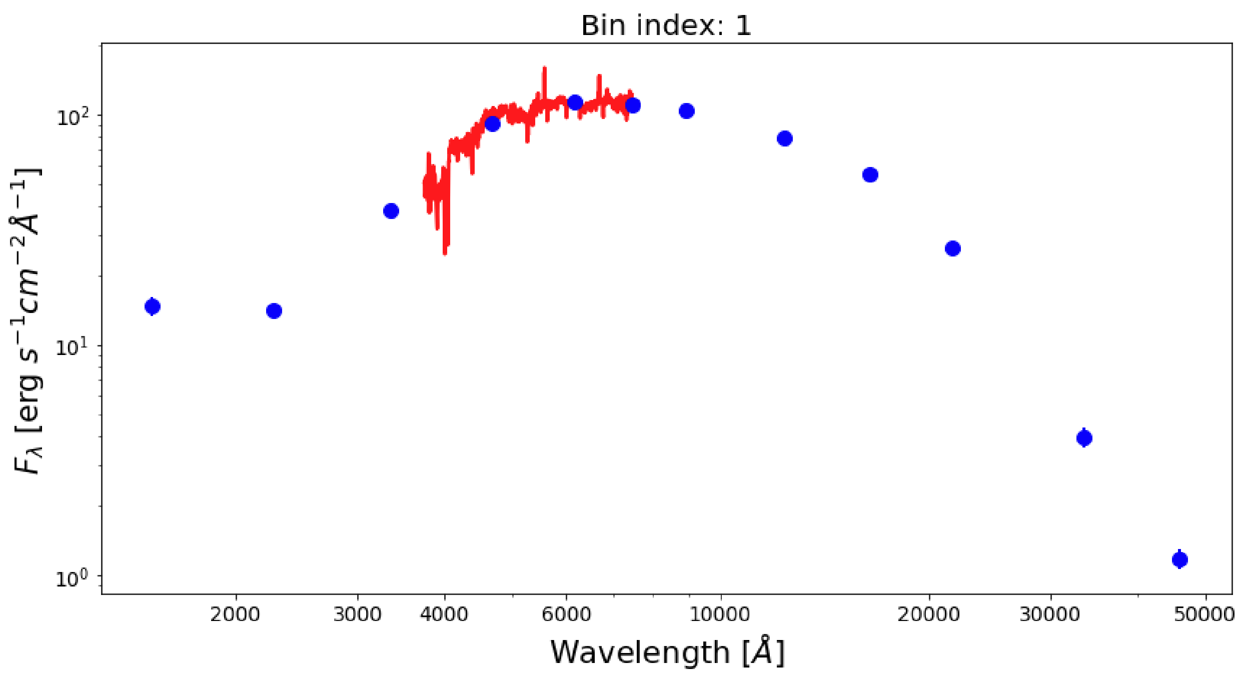
Now, let's plot SEDs of four examples of spatial bins, three with spectrophotometric data and one with photometric only data.

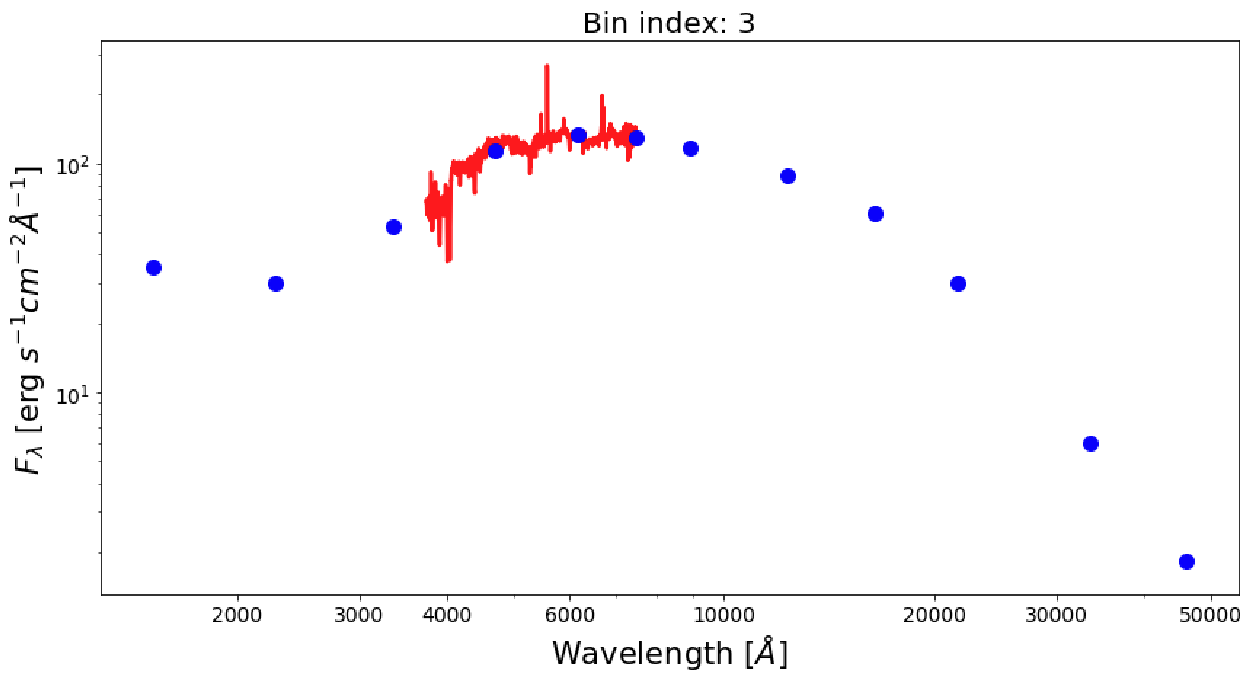
```
from matplotlib.ticker import ScalarFormatter

bin_ids = [1, 3, 8, 30]
for ii in range(0,len(bin_ids)):

        fig1 = plt.figure(figsize=(14,7))
        f1 = plt.subplot()
```

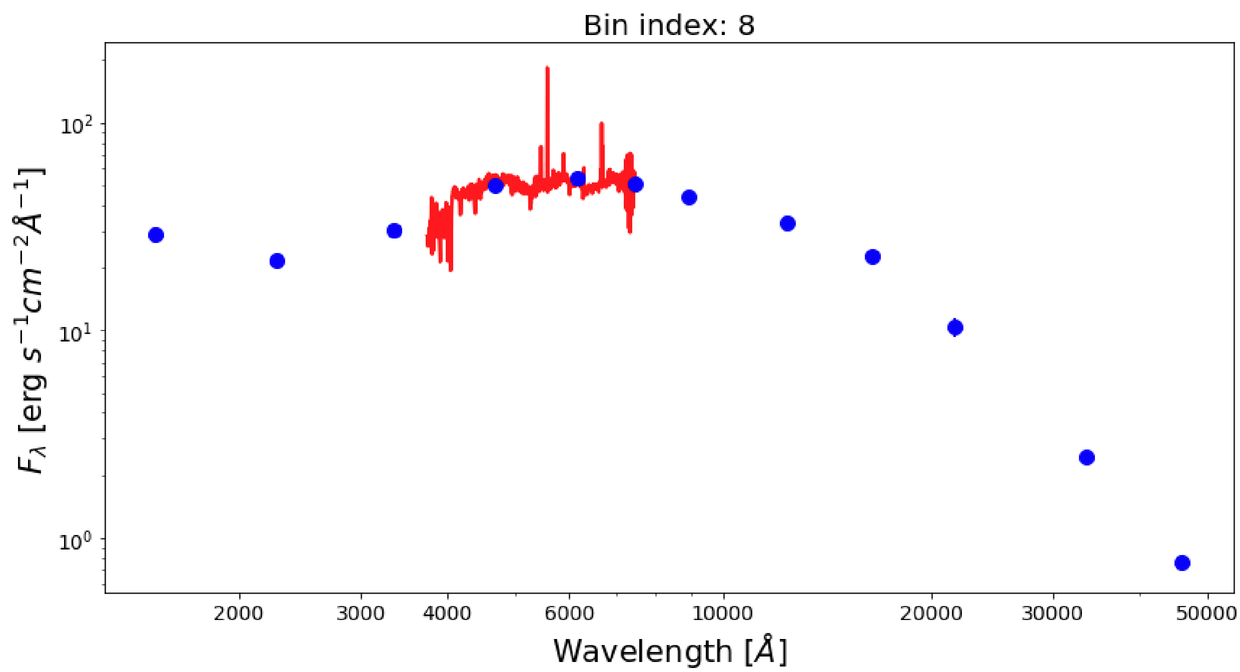
(continues on next page)

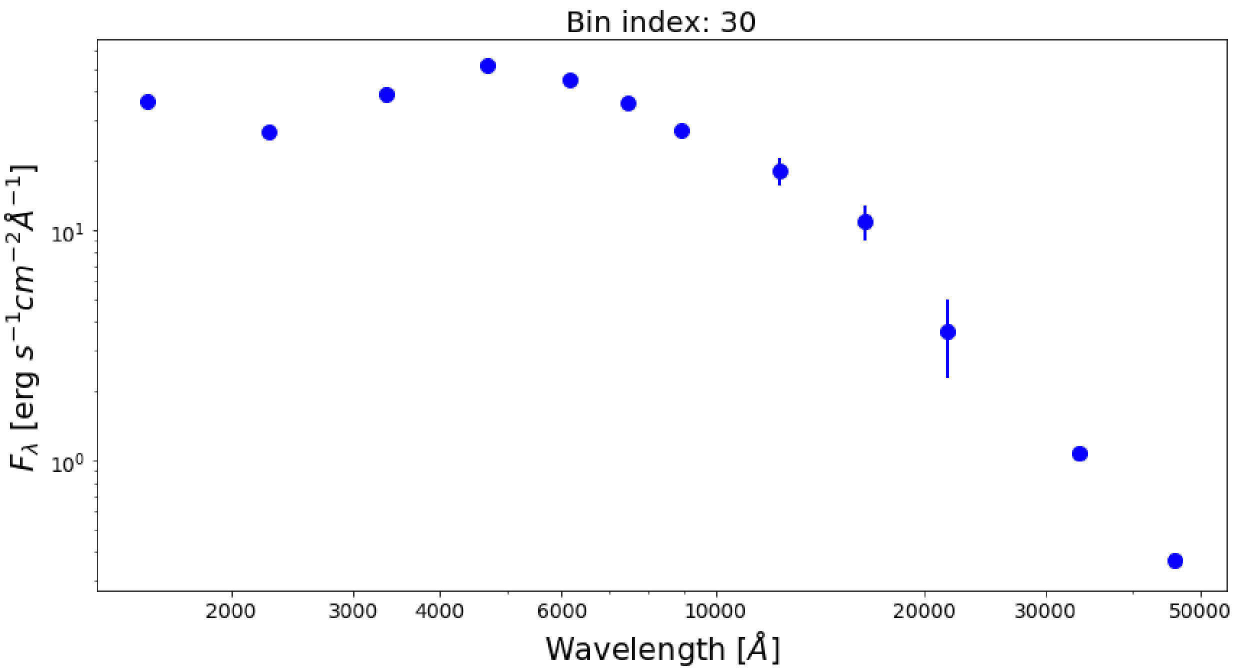
(continued from previous page)

```
        plt.title("Bin index: %d" % bin_ids[ii], fontsize=20)
        f1.set_yscale('log')
        f1.set_xscale('log')
        plt.setp(f1.get_yticklabels(), fontsize=14)
        plt.setp(f1.get_xticklabels(), fontsize=14)
        plt.xlabel(r'Wavelength $[\AA]$', fontsize=21)
        plt.ylabel(r'$F_{\lambda}$ [erg $s^{-1}cm^{-2}\AA^{-1}$]', fontsize=21)
        xticks = [2000,3000,4000,6000,10000,20000,30000,50000]
        plt.xticks(xticks)
        for axis in [f1.xaxis]:
                axis.set_major_formatter(ScalarFormatter())
        if np.sum(bin_spec_flux[int(bin_ids[ii])-1])>0:
                plt.plot(spec_wave[20:nwaves-20], bin_spec_flux[int(bin_
↪ids[ii])-1][20:nwaves-20],
                                lw=2.5, color='red')

        plt.errorbar(photo_wave, bin_photo_flux[int(bin_ids[ii])-1],
                        yerr=bin_photo_flux_err[int(bin_ids[ii])-1],␣
↪markersize=10,
                        color='blue', fmt='o', lw=2)
```




Bin index: 3
10^2
10^1
F_λ [erg $s^{-1}cm^{-2}\AA^{-1}$]
2000
3000
4000
6000
10000
20000
30000
50000
Wavelength [Å]


Bin index: 8
10^2
10^1
10^0
F_λ [erg $s^{-1}cm^{-2}\AA^{-1}$]
2000
3000
4000
6000
10000
20000
30000
50000
Wavelength [Å]



1.9.2 Pixel binning on images

(This section is still under construction…)

1.10 SED fitting

piXedfit provides independent SED fitting functions that are collected in `piXedfit_fitting` module. The input SED can be either integrated SEDs of galaxies that are fit individually (i.e., one-by-one) or spatially resolved SEDs of a galaxy that are read from binned data cube. For an input of spectrophotometric SED, the SED fitting module can fit simultaneously the photometric SED and the spectrum, providing a powerful constraint for breaking the age–dust–metallicity degeneracies.

1.10.1 Free parameters

Before delving into the SED fitting functions, let's see the list of free parameters in the SED fitting. A complete list of parameters and their meaning are given in Abdurro'uf et al. (2021) (Table 1 therein). Each parameter is given with a unique name (i.e., keyword) which also represents how it is sampled in the SED fitting process. Most of the parameters are sampled in logarithmic scale. The keywords of the parameters (with the same order as they appear in the Table 1 of the above paper) are the following: *log_mass*, *logzsol*, *log_age*, *log_tau*, *log_t0*, *log_alpha*, *log_beta*, *dust1*, *dust2*, *dust_index*, *gas_logu*, *log_umin*, *log_gamma*, *log_qpah*, *log_fagn*, and *log_tauagn*. Among the those parameters, only *gas_logu* (ionization parameter in the nebular emissin modeling) is not free in the current version of piXedfit, instead it is fixed in generating model and SED fitting. Parameters that have keyword started with *log* are sampled in logarithmic scale. Only three parameters, *dust1*, *dust2*, and *dust_index*, are sampled in a linear scale.

Note: Please note that the number of free parameters involved in the SED fitting process is not determined when we configure the SED fitting functions, instead they are determined when we generate model spectral templates (described below).

1.10.2 Generating model spectra at rest-frame

Before performing SED fitting, we need to generate model spectral templates that are calculated at rest-frame of the galaxy (i.e., independent of redshift). Please see the description and tutorial for detailed information on how to generate this model spectra.

Note: It is important to note that all the configurations in the SED modeling and SED fitting are determined in this step.

Here we determine what types of models that we want to generate and fit to our observed SEDs. This modeling configuration include the choice of initial mass function (IMF; *imf_type*), the choice of star formation history (SFH; *sfh_form*), the choice of dust attenuation law (*dust_law*), whether to switch on/off the following features (nebular emission *add_neb_emission*, dust emission *duste_switch*, AGN dusty torus emission *add_agn*, intergalactic medium *add_igm_absorption*). Those features have parameters associated with them. This determines the free parameters that will be involved in the SED fitting process. Please see the API reference here for more detailed information.

The model parameters are randomly drawn but uniformly distributed within the chosen ranges. This process will generate a set of model spectra that are stored in HDF5 file format.

Note: It is not necessary to generate a set of model spectra for each galaxy in our sample.

1.10.3 Fitting methods and some tips

Overall, piXedfit adopts Bayesian statistics in the SED fitting process. It provides two options for sampling the posterior probability distribution: Markov Chain Monte Carlo (MCMC) and random dense sampling of parameter space (RDSPS). Please refer to Abdurro'uf et al. (2021) for more detailed information on these two fitting methods. While MCMC is computationally intensive, it is recommended to use this method whenever possible (e.g., sufficient computational resource is available).

Warning: While the RDSPS method is quite fast (few seconds per SED), this method is very sensitive to our priors (described below). One need to make sure that the selected ranges of parameters are sufficiently wide, especially the parameter age (*log_age*). The derived stellar mass (*log_mass*) from RDSPS fitting is sensitive to the adopted range of *log_age*.

Some tips: * Generate multiple sets of model spectra (i.e., multiple HDF5 files) for sample of galaxies that have wide range of redshift. * For the stellar age parameter (*log_age*) to be sufficiently sampled, it is recommended to set a range for *log_age* with minimum value of -1.0 or -2.0 and maximum value that corresponds to the age of the universe at the redshift of the target galaxy. For larger sample of galaxies, one can produce multiple sets of model spectra with various maximum ages. * Since the RDSPS fitting is very sensitive to our priors, it is recommended to use physically motivated priors. This can be in the forms of joint priors among parameters, such as mass-metallicity and mass-age priors that are cnstructed based on empirical scaling relations.

1.10.4 Defining priors

Priors for SED fitting can be defined using the *piXedfit.piXedfit_fitting.priors* class. First, we choose ranges of parameters (no need to define it for all parameters). Then, we can choose forms of the priors. There are several forms available, including uniform, Gaussian function, Gamma function, and Student's t function. There is also a choice for defining arbitrary form if users have their own prior form. It is also possible to define joint prior between stellar mass and a particular parameter.

Please see the API reference here for more information on defining priors.

1.10.5 Fitting individual SED

If one has a single SED or collection of SEDs and intend to fit the SED individually one-by-one, there are two functions available: *piXedfit.piXedfit_fitting.singleSEDfit()* and `piXedfit.piXedfit_fitting.singleSEDfit_specphoto()`. The former is designed from input photometric SED, while the latter is for input spectrophotometric SED. Please see the API references here and here for more information about these functions.

The following is an example of script for fitting a single photometric SED.

```
from piXedfit.piXedfit_fitting import singleSEDfit

# input SED to be fit
obs_flux = [1.4638932316652199e-16, 1.4038745561641324e-16, 3.
↪8179726118818497e-16, 9.173751654543811e-16,
        1.12950610607557e-15, 1.1081589447235416e-15, 1.0348753094599238e-15,␣
↪7.921053538416444e-16,
        5.479813400644683e-16, 2.620524286818637e-16, 3.9683859472674366e-17,␣
↪1.1746973239557648e-17]
obs_flux_err = [1.3245499396855221e-17, 5.1672408967693435e-18, 1.
↪0684115183145074e-17, 2.8758039375671638e-18,
        2.4362783664430513e-18, 2.43151290865674e-18, 6.020149814200945e-18, 1.
↪4838739345484332e-17,
        1.1592521846969518e-17, 8.156051820519197e-18, 4.044010707857459e-18,␣
↪1.104563357898008e-18]
filters = ['galex_fuv', 'galex_nuv', 'sdss_u', 'sdss_g', 'sdss_r', 'sdss_i',
↪'sdss_z', '2mass_j',
        '2mass_h', '2mass_k', 'wise_w1', 'wise_w2']

# redshift
gal_z = 0.0188977

# Name of HDF5 file containing model spectra at rest-frame
models_spec = "s_cb_dpl_cf_nde_na_100k.hdf5"

# call function for defining priors in SED fitting
from piXedfit.piXedfit_fitting import priors

# define ranges of some parameters
ranges = {'logzsol':[-2.0,0.2], 'dust1':[0.0,3.0], 'dust2':[0.0,3.0], 'log_age
↪':[-1.0,1.14]}
pr = priors(ranges)
params_ranges = pr.params_ranges()
```

(continues on next page)

(continued from previous page)

```
# define prior forms
prior1 = pr.uniform('logzsol')
prior2 = pr.uniform('dust1')
prior3 = pr.uniform('dust2')
prior4 = pr.uniform('log_age')
params_priors = [prior1, prior2, prior3, prior4]

# choice of SED fitting method
fit_method = 'mcmc'

nproc = 20           # number of cores to be used in the calculation

# ouptut name
name_out_fits = "fitting.fits"
singleSEDfit(obs_flux, obs_flux_err, filters, models_spec, params_
↪ranges=params_ranges, params_priors=params_priors,
                        fit_method=fit_method, gal_z=gal_z, nwalkers=100,␣
↪nsteps=600, nproc=proc, initfit_nmodels_mcmc=100000,
                        store_full_samplers=1, name_out_fits=name_out_fits)
```

1.10.6 Fitting spatially resolved SEDs

For fitting spatially resolved SEDs in the binned data cube, one can use *piXedfit.piXedfit_fitting.SEDfit_from_binmap()* and `piXedfit.piXedfit_fitting.SEDfit_from_binmap_specphoto()` functions. The former is designed for photometric data cube, while the latter is for spectrophotometric data cube. Please see the API references of these functions for more information.

The following is an example of script for performing SED fitting to spectrophotometric data cube. In this script, *binid_range* determines the spatial bins that are going to be fit. With this, we can separate calculations into multiple terminals (i.e., processes) on a multi cores cluster or super computer. Please note that each script also uses multiple cores in parallel computation.

```
from piXedfit.piXedfit_fitting import SEDfit_from_binmap_specphoto

fits_binmap = "pixbin_corr_specphoto_fluxmap_ngc309.fits"
binid_range = [0,20]       # range for the ids of spatial bins to be fit

# Name of HDF5 file containing model spectra at rest-frame
models_spec = "s_cb_dpl_cf_nde_na_100k.hdf5"

# call function for defining priors in SED fitting
from piXedfit.piXedfit_fitting import priors

# define ranges of some parameters
ranges = {'logzsol':[-2.0,0.2], 'dust1':[0.0,3.0], 'dust2':[0.0,3.0], 'log_age
↪':[-1.0,1.14]}
pr = priors(ranges)
params_ranges = pr.params_ranges()

# define prior forms
prior1 = pr.uniform('logzsol')
```

(continues on next page)

(continued from previous page)

```
prior2 = pr.uniform('dust1')
prior3 = pr.uniform('dust2')
prior4 = pr.uniform('log_age')
params_priors = [prior1, prior2, prior3, prior4]

# choice of SED fitting method
fit_method = 'mcmc'

# range of wavelength in which the spectrum will be fit
wavelength_range = [3000,9000]

# spectral resolution in Angstrom
spec_sigma = 3.5     # spectral resolution of MaNGA

nproc = 20           # number of cores to be used in the calculation
SEDfit_from_binmap_specphoto(fits_binmap, binid_range=binid_range, models_
↪spec=models_spec,
                                wavelength_range=wavelength_range,params_
↪ranges=params_ranges,
                                params_priors=params_priors,fit_method=fit_
↪method,spec_sigma=spec_sigma,
                                nwalkers=100,nsteps=800,nproc=nproc,initfit_
↪nmodels_mcmc=100000,
                                store_full_samplers=1)
```

1.11 Analyzing fitting results

To check SED fitting results, piXedfit provides functions in `piXedfit_analysis` module for making visualization plots. There are three kinds of plots that can be made using this module.

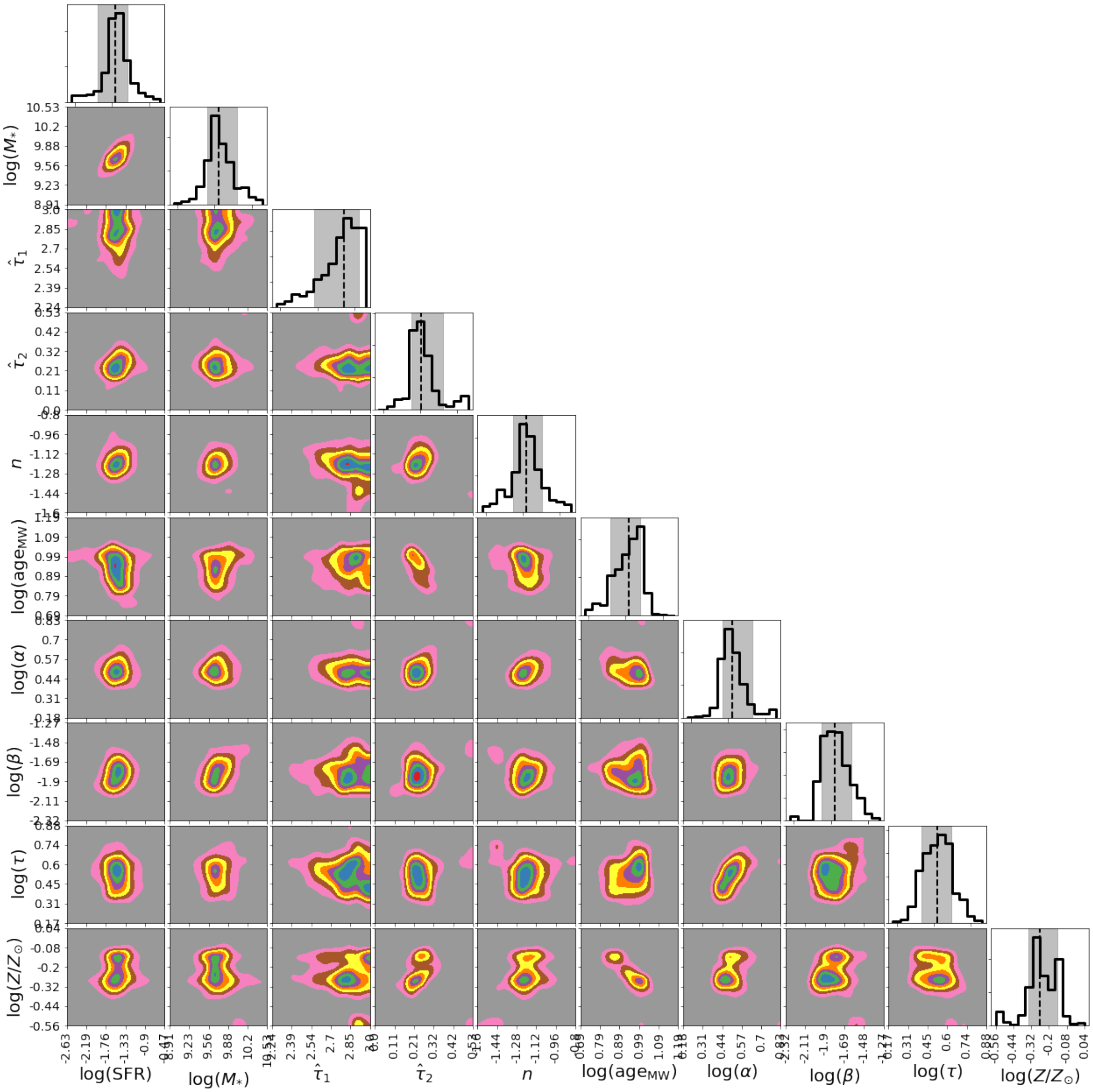
- **Corner plot**, which shows joint posterior distributions of the parameters. This plot can be made using *piXedfit.piXedfit_analysis.plot_corner()*
- **SED plot**, which shows best-fit model SED. This plot can be made using *piXedfit.piXedfit_analysis.plot_SED()* function.
- **Star formation history (SFH) plot**, which shows reconstructed SFH. This plot can be made using *piXedfit.piXedfit_analysis.plot_sfh_mcmc()*. This plot can only be made for result of MCMC fitting with the current version of piXedfit.

Please see the API references for more information about those functions. Below, we will demonstrate how to produce such plots for an example of fitting result to a spectrophotometric SED.

1.11.1 Corner plot

```
from piXedfit.piXedfit_analysis import plot_corner

params=['log_sfr', 'log_mass', 'dust1', 'dust2', 'dust_index', 'log_mw_age',
        'log_alpha', 'log_beta', 'log_tau', 'logzsol']
label_params={'dust1': '$\\hat \\tau_{1}$', 'dust2': '$\\hat \\tau_{2}$',
        'dust_index': '$n$', 'log_alpha': 'log($\\alpha$)', 'log_beta': 'log($\
→\beta$)',
        'log_mass': 'log($M_{*}$)', 'log_mw_age': 'log($\\mathrm{age}_{\\mathrm
→{MW}}$)',
        'log_sfr': 'log(SFR)', 'log_t0': 'log($t_{0}$)', 'log_tau': 'log($\\tau
→$)',
        'logzsol': 'log($Z/Z_{\\odot}$)'}

name_sampler_fits = "mcmc_bin1.fits"
plot_corner(name_sampler_fits, params=params, label_params=label_params)
```

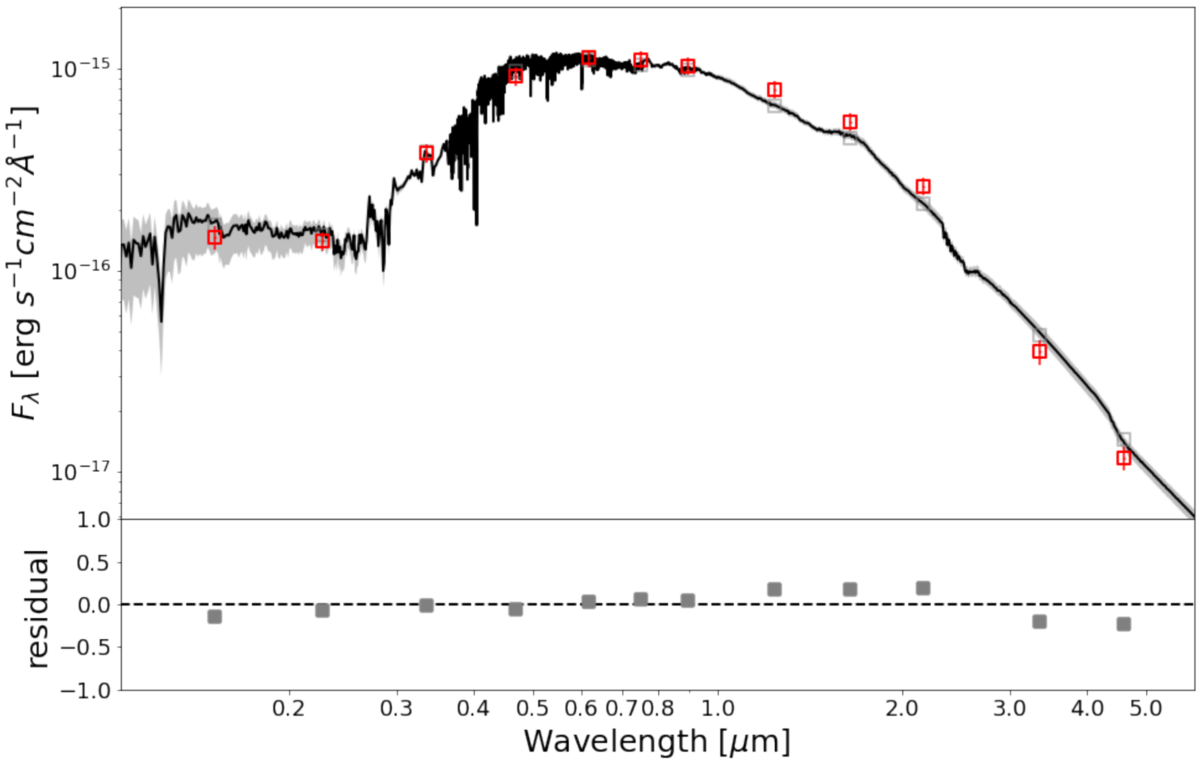
We will get a corner plot as shown below.



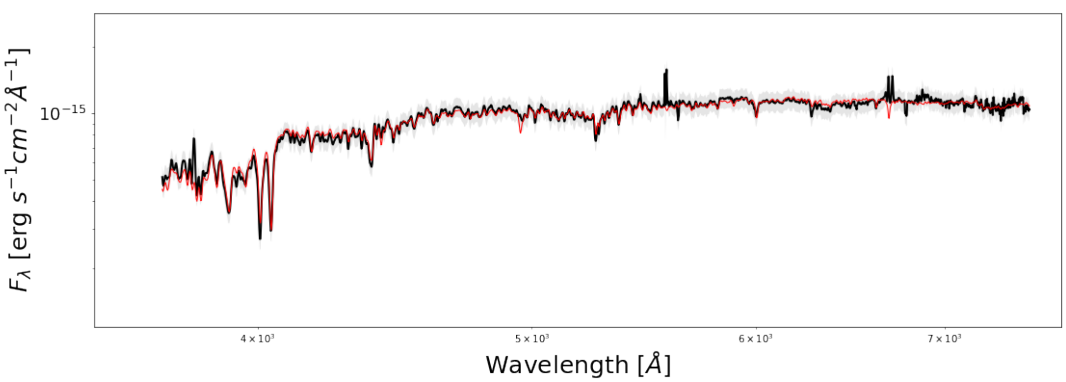
1.11.2 SED plot

```
from piXedfit.piXedfit_analysis import plot_SED

name_sampler_fits = "mcmc_bin1.fits"
plot_SED(name_sampler_fits, decompose=0,
        xticks=[0.2,0.3,0.4,0.5,0.6,0.7,0.8,1.0,2.0,3.0,4.0,5.0])
```

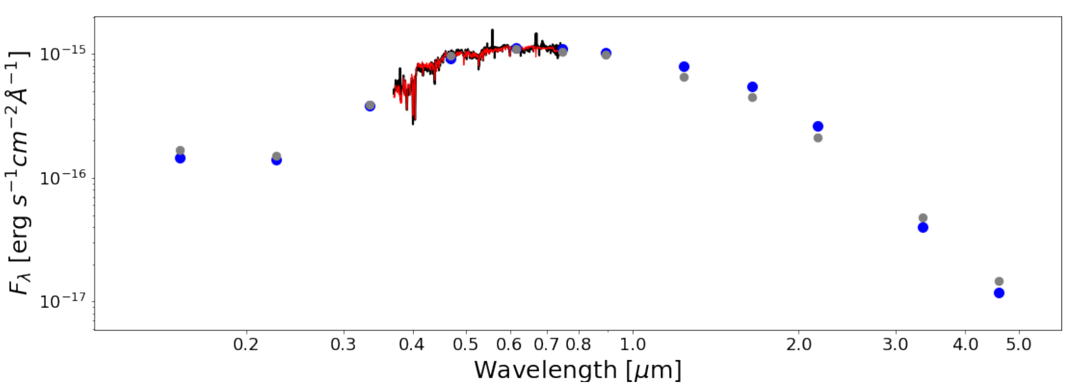
We will get SED plot as shon below.



F_λ [erg $s^{-1} cm^{-2} \AA^{-1}$]
10^{-15}
10^{-16}
10^{-17}
residual
1.0
0.5
0.0
−0.5
−1.0
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
2.0
3.0
4.0
5.0
Wavelength [μm]



F_λ [erg $s^{-1} cm^{-2} \AA^{-1}$]
10^{-15}
Wavelength [$\AA$]

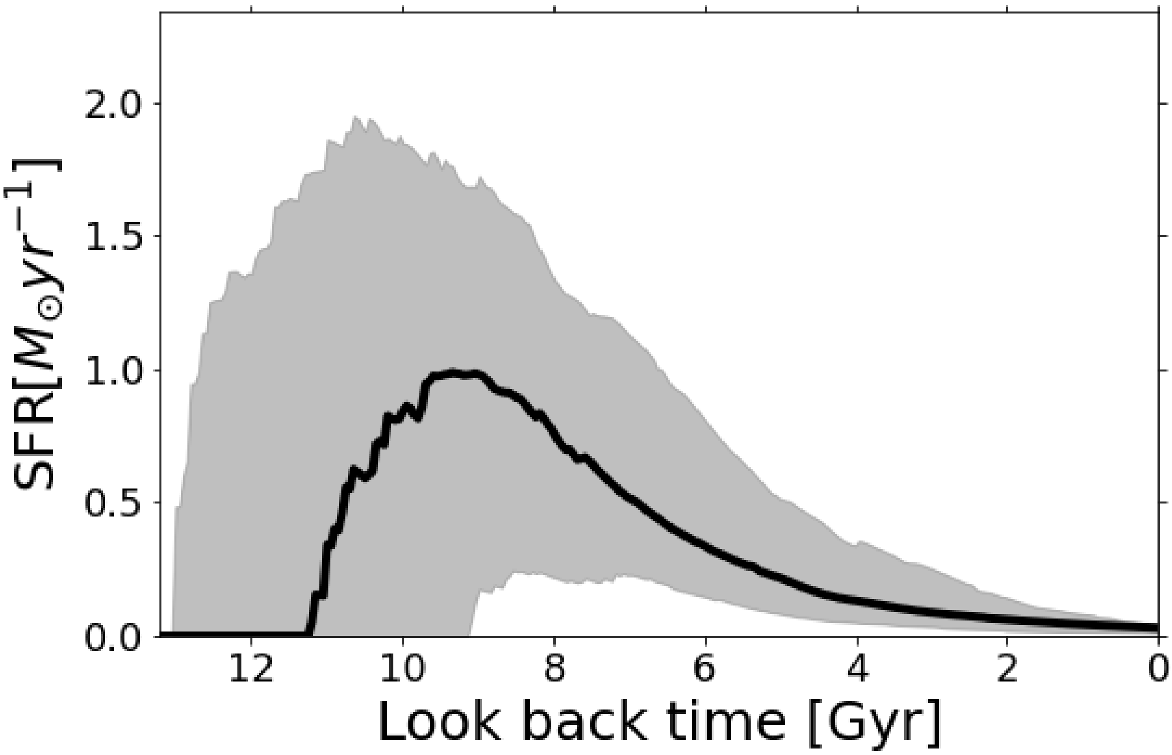


F_λ [erg $s^{-1} cm^{-2} \AA^{-1}$]
10^{-15}
10^{-16}
10^{-17}
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1.0
2.0
3.0
4.0
5.0
Wavelength [μm]

1.11.3 SFH plot

```
from piXedfit.piXedfit_analysis import plot_sfh_mcmc

name_sampler_fits = "mcmc_bin1.fits"
plot_sfh_mcmc(name_sampler_fits)
plt.show()
```

We will get SFH plot as shown below.



1.12 Get maps of spatially resolved properties

Once we finished SED fitting to all spatial bins in the galaxy, we can construct maps of properties from the collection of those fitting results (in FITS files). For this, we can use *piXedfit.piXedfit_fitting.maps_parameters()* function in the piXedfit_fitting module. Please see the API reference here for more information about this function. An example of script to get the maps properties is shown below.

```
from piXedfit.piXedfit_fitting import maps_parameters

fits_binmap = "pixbin_corr_specphoto_fluxmap_ngc309.fits"
hdu = fits.open(fits_binmap)
nbins = int(hdu[0].header['NBINSPH'])
hdu.close()

bin_ids = []
name_sampler_fits = []
```

(continues on next page)

(continued from previous page)

```
for ii in range(0,nbins):
        bin_ids.append(ii)
        name = "mcmc_bin%d.fits" % (ii+1)
        name_sampler_fits.append(name)

fits_fluxmap = "corr_specphoto_fluxmap_ngc309.fits"
name_out_fits = "maps_properties.fits"
maps_parameters(fits_binmap=fits_binmap, bin_ids=bin_ids, name_sampler_
↪fits=name_sampler_fits,
                fits_fluxmap=fits_fluxmap,refband_SFR=0, refband_SM=11, name_
↪out_fits=name_out_fits)
```

We can then make plots from the maps stored in the output FITS file.

```
maps = fits.open("maps_properties.fits")
gal_region = maps['galaxy_region'].data

# calculate physical size of a single pixel
from piXedfit.piXedfit_images import kpc_per_pixel

z = float(maps[0].header['gal_z'])
arcsec_per_pixel = 1.5                          # pixel size in arcsec
kpc_per_pix = kpc_per_pixel(z,arcsec_per_pixel)

rows, cols = np.where(gal_region == 0) # get pixels outside the galaxy's region␣
↪of interest

# stellar mass surface density
map_prop_SM = np.log10(np.power(10.0,maps['pix-log_mass-p50'].data)/kpc_per_
↪pix/kpc_per_pix)
map_prop_SM[rows,cols] = float('nan')

# surface density of SFR
map_SFR = np.log10(np.power(10.0,maps['pix-log_sfr-p50'].data)/kpc_per_pix/kpc_
↪per_pix)
map_SFR[rows,cols] = float('nan')

# Av dust attenuation
map_Av = 1.086*maps['bin-dust2-p50'].data
map_Av[rows,cols] = float('nan')

# stellar metallicity
map_logzsol  = maps['bin-logzsol-p50'].data
map_logzsol[rows,cols] = float('nan')

# Mass-weighted age
map_mw_age  = np.power(10.0,maps['bin-log_mw_age-p50'].data)
map_mw_age[rows,cols] = float('nan')

# plotting
from mpl_toolkits.axes_grid1 import make_axes_locatable
```

(continues on next page)

(continued from previous page)

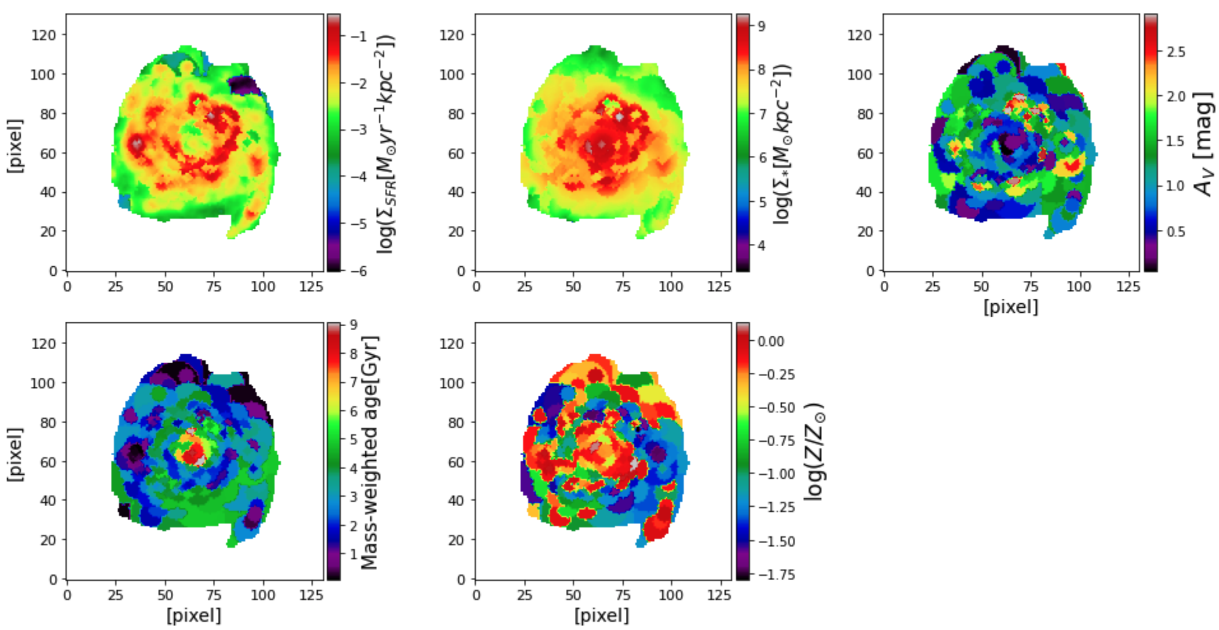
```
fig1 = plt.figure(figsize=(14,7))

###=> SFR surface density
f1 = fig1.add_subplot(2, 3, 1)
plt.ylabel('[pixel]', fontsize=14)
plt.setp(f1.get_xticklabels(), fontsize=11)
plt.setp(f1.get_yticklabels(), fontsize=11)

plt.imshow(map_SFR, origin='lower', cmap='nipy_spectral')
ax = plt.gca()
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cb = plt.colorbar(cax=cax)
cb.set_label(r'log($\Sigma_{SFR}[M_{\odot}yr^{-1}kpc^{-2}]$)', fontsize=15)


###=> stellar mass surface density
f1 = fig1.add_subplot(2, 3, 2)
plt.setp(f1.get_xticklabels(), fontsize=11)
plt.setp(f1.get_yticklabels(), fontsize=11)

plt.imshow(map_prop_SM, origin='lower', cmap='nipy_spectral')
ax = plt.gca()
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cb = plt.colorbar(cax=cax)
cb.set_label(r'log($\Sigma_{*}[M_{\odot}kpc^{-2}]$)', fontsize=15)

###=> AV dust attenuation
f1 = fig1.add_subplot(2, 3, 3)
plt.xlabel('[pixel]', fontsize=14)
plt.setp(f1.get_xticklabels(), fontsize=11)
plt.setp(f1.get_yticklabels(), fontsize=11)

plt.imshow(map_Av, origin='lower', cmap='nipy_spectral')
ax = plt.gca()
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cb = plt.colorbar(cax=cax)
cb.set_label(r'$A_{V}$ [mag]', fontsize=18)

### mass-weighted age
f1 = fig1.add_subplot(2, 3, 4)
plt.xlabel('[pixel]', fontsize=14)
plt.ylabel('[pixel]', fontsize=14)
plt.setp(f1.get_xticklabels(), fontsize=11)
plt.setp(f1.get_yticklabels(), fontsize=11)

plt.imshow(map_mw_age, origin='lower', cmap='nipy_spectral')
ax = plt.gca()
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
```

(continues on next page)

(continued from previous page)

```
cb = plt.colorbar(cax=cax)
cb.set_label('Mass-weighted age[Gyr]', fontsize=15)

### stellar metallicity
f1 = fig1.add_subplot(2, 3, 5)
plt.xlabel('[pixel]', fontsize=14)
plt.setp(f1.get_xticklabels(), fontsize=11)
plt.setp(f1.get_yticklabels(), fontsize=11)

plt.imshow(map_logzsol, origin='lower', cmap='nipy_spectral')
ax = plt.gca()
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cb = plt.colorbar(cax=cax)
cb.set_label(r'log($Z/Z_{\odot}$)', fontsize=17)

plt.subplots_adjust(left=0.07, right=0.95, bottom=0.1, top=0.98, hspace=0.2,␣
↪wspace=0.3)
plt.show()
```

We can then get a plot as shown below.



1.13 Tutorials

Some tutorials are provided in the form of jupyter notebooks and can be found in the github repository. Those tutorials demonstrate the analyses from raw data up to the SED fitting and retrieving maps of the properties of stellar population and dust. We classified the tutorials based on the wavelength range covered by the data sets.

1.13.1 Analysis of FUV–FIR imaging data

The tutorial on this analysis can be found here.

1.13.2 Analysis of FUV–NIR imaging data combined with IFS data

- With IFS data from the MaNGA survey: FUVtoNIR_MaNGA
- With iFS data from the CALIFA survey: FUVtoNIR_CALIFA

1.14 Pixel binning

Please see the pixel binning animation here.

1.15 SED fitting

Please see the SED fitting animations here.

1.16 piXedfit_images

`piXedfit.piXedfit_images.`**`EBV_foreground_dust`**`(`*ra*, *dec*`)`

Function for estimating E(B-V) dust attenuation due to the foreground Galactic dust attenuation at a given coordinate on the sky.

Parameters

- **`ra`** – Right ascension coordinate in degree.
- **`dec`** – Declination coordinate in degree.

Returns ebv

E(B-V) value.

`piXedfit.piXedfit_images.`**`calc_pixsize`**`(`*fits_image*`)`

Function to get pixel size of an image

Parameters

`fits_image` – Input image.

Returns pixsize_arcsec

Pixel size in arc second.

piXedfit.piXedfit_images.**central_brightest_pixel**(*flux_maps_fits*, *filter_id*, *xrange=None*, *yrange=None*)

Function to get the central brightest pixel within a map of flux in a particular band.

Parameters

- **flux_maps_fits** – Input FITS file of the flux maps produced from the image processing.
- **filter_id** – The filter of the image where the central pixel is to be found.
- **xrange** – Range in x-axis for searching area.
- **yrange** – Range in y-axis for searching area.

piXedfit.piXedfit_images.**convert_flux_unit**(*wave*, *flux*, *init_unit='erg/s/cm2/A'*, *final_unit='Jy'*)

Function for converting flux unit

Parameters

- **wave** – Wavelength grids.
- **flux** – Flux grids.
- **init_unit** – Initial unit of the input fluxes. Options are: 'erg/s/cm2/A', 'erg/s/cm2', 'Jy', 'uJy'.
- **final_unit** – Final unit for conversion. Options are: 'erg/s/cm2/A', 'erg/s/cm2', 'Jy', 'uJy'.

piXedfit.piXedfit_images.**create_psf_matching_kernel**(*init_PSF_name*, *target_PSF_name*, *pixscale_init_PSF*, *pixscale_target_PSF*, *window='top_hat'*, *window_arg=1.0*)

A function for creating convolution kernel for PSF matching given initial and target PSFs.

Parameters

- **init_PSF_name** – Image of input/initial PSF.
- **target_PSF_name** – Image of target PSF.
- **pixscale_init_PSF** – Pixel size of the initial PSF in arcsec.
- **pixscale_target_PSF** – Pixel size of the target PSF in arcsec.
- **window** – Options are 'top_hat' and 'cosine_bell'.
- **window_arg** – Coefficient value of the window function, following Photutils.
- **kernel** – The data of convolution kernel in 2D array.

piXedfit.piXedfit_images.**crop_stars_galregion_fits**(*input_fits*, *x_cent*, *y_cent*, *radius*, *name_out_fits=None*)

Function for cropping foreground stars within a galaxy's region of interst.

Parameters

- **input_fits** – The input FITS file containing the data cube that is produced using the `flux_map()` funciton in the `images_processing` class.
- **x_cent** – 1D array of x coordinates of the center of the stars.
- **y_cent** – 1D array of y coordinates of the center of the stars.
- **radius** – 1D array of the estimated circular radii of the stars.
- **name_out_fits** – Desired name for the output FITS file. If None, a generic name will be made.

piXedfit.piXedfit_images.**draw_aperture_on_maps_fluxes**(*flux_maps_fits*, *ncols=6*, *e=[0.0]*, *pa=[45.0]*, *cent_x=None*, *cent_y=None*, *radius=[5.0]*, *colors=None*, *lw=3*, *savefig=True*, *name_plot=None*)

Function for drawing apertures on top of the multiband fluxes maps. This function can plot more than one aperture.

Parameters

- **flux_maps_fits** – input FITS file containing the maps of fluxes produced from the image processing.
- **e** – Ellipticity of the apertures. Input in list format if want to make multiple apertures.
- **pa** – Position angle of the elliptical apertures. Input in list format if want to make multiple apertures.
- **cent_x** – The x coordinate of the central pixel. Input in list format if want to make multiple apertures.
- **cent_y** – The y coordinate of the central pixel. Input in list format if want to make multiple apertures.
- **radius** – The radius of the aperture. Input in list format if want to make multiple apertures.
- **colors** – Colors of the apertures. Input in list format if want to make multiple apertures.
- **lw** – Line width of the apertures in the plot.
- **savefig** – Decide whether to save the plot or not.
- **name_plot** – Name for the plot.

piXedfit.piXedfit_images.**get_pixels_SED_fluxmap**(*flux_maps_fits*, *pix_x=None*, *pix_y=None*, *all_pixels=False*)

Function to get SEDs of pixels.

Parameters

- **flux_maps_fits** – Input FITS file of the multiband fluxes.
- **pix_x** – One dimensional array of x coordinates. Only relevant if all_pixels=False.
- **pix_y** – One dimensional array of the y coordinates. Only relevant if all_pixels=False.
- **all_pixels** – (optional) An option to get SEDs of all pixels.

Returns pix_x
x coordinates.

Returns pix_y
y coordinates.

Returns pix_SED_flux
Fluxes of the pixels: (npixs,nbands)

Returns pix_SED_flux_err
Flux uncertainties of the pixels: (npixels,nbands)

Returns photo_wave
Central wavelength of the filters.

piXedfit.piXedfit_images.**get_total_SED**(*flux_maps_fits*)

Function to calculate total (i.e., integrated) SED from input maps of multiband fluxes.

Parameters

flux_maps_fits – Input FITS file containing maps of fluxes produced from the image processing.

Returns tot_SED_flux

Total fluxes in multiple bands. The format is 1D array.

Returns tot_SED_flux_err

Total flux uncertainties in multiple bands. The format is 1D array.

Returns photo_wave

The central wavelength of the filters.

class piXedfit.piXedfit_images.**images_processing**(*filters*, *sci_img*, *var_img*, *gal_ra*, *gal_dec*, *dir_images=None*, *img_unit=None*, *img_scale=None*, *img_pixsizes=None*, *run_image_processing=True*, *flag_psfmatch=0*, *flag_reproject=0*, *flag_crop=0*, *kernels=None*, *gal_z=None*, *stamp_size=[101, 101]*, *remove_files=True*, *idfil_align=None*)

A Python class for processing multiband imaging data and producing a data cube containing maps of multiband fluxes that are matched in spatial resolution and sampling. The image processing basically includes the PSF matching to homogenize the spatial resolution of the multiband imaging data and spatial resampling and reprojection to homogenize the spatial sampling (i.e., pixel size) and spatil reprojection of the mulltiband imaging data. A list of imaging data sets that can be handled automatically by this class in the current version of piXedfit can be seen at *List of imaging data*. However, one need to download convolution kernels from this link and put those inside data/kernels/ within the piXedfit directory ($PIXEDFIT_HOME/data/kernels). These kernels are not included in the piXedfit repository because of the large file sizes. This class can also handle other imaging data. For this, one need to provide kernels for PSF matching. These kernel images should have the same pixel size as the corresponding input images.

Parameters

- **filters** – List of photometric filters names. To check the filters currently available and manage them (e.g., adding new ones) please see this page. For this input, it is not mandatory to make the input filters in a wavelength order.
- **sci_img** – Dictionary containing names of the science images. An example of input: sci_img={'sdss_u':'img1.fits', 'sdss_g':'img2.fits'}
- **var_img** – Dictionary containing names of the variance images. It has a similar format to sci_img.
- **gal_ra** – Right Ascension (RA) coordinate of the target galaxy. This should be in degree.
- **gal_dec** – Declination (DEC) coordinate of the target galaxy. This should be in degree.
- **dir_images** – Directory where images are stored.
- **img_unit** – (optional) Unit of pixel value in the multiband images. The acceptable format of this input is a python dictionary, similar to that of sci_img. This input is optional. This input will only be considered (and required) if the input images are not among the default list of recognized imaging data in piXedfit (i.e. GALEX, SDSS, 2MASS, WISE, Spitzer, and Herschel). The allowed units are: (1)"erg/s/cm2/A", (2) "Jy", and (3) "MJy/sr".
- **img_scale** – (optional) Scale of the pixel value with respect to the unit in img_unit. For instance, if image is in unit of MJy, the img_unit can be set to be "Jy" and img_scale is set

to be 1e+6. This input is only relevant if the input images are not among the default list of recognized images in piXedfit. The format of this input should be in python dictionary, similar to sci_img.

- **img_pixsizes** – (optional) Pixel sizes (in arcsecond) of the input imaging data. This input should be in dictionary format, similar to sci_img. If not provided, pixel size will be calculated based on the WCS information in the header of the FITS file.
- **flag_psfmatch** – (optional) Flag stating whether the multiband imaging data have been PSF-matched or not. The options are: (1) 0 means hasn't been PSF-matched, and (2)1 means has been PSF-matched.
- **flag_reproject** – (optional) Flag stating whether the multiband imaging data have been spatially-resampled and matched in the projection. The options are: (1)0 means not yet, and (2)1 means has been carried out.
- **flag_crop** – (optional) Flag stating whether the multiband imaging data have been cropped around the target galaxy. The options are: (1)0 means not yet, and (2)1 means has been cropped. If flag_crop=0, cropping will be done according to the input stamp_size. If flag_crop=1, cropping will not be done.
- **kernels** – (optional) Dictionary containing names of FITS files for the kernels to be used for the PSF matching process. If None, the code will first look for kernels inside piXedfit/data/kernels directory. If an appropriate kernel is not found, the kernel input remain None and PSF matching will not be done for the corresponding image. If external kerenels are avaiable, the kernel images should have the same pixel size as the corresponding input images and the this input should be in a dictionary format, which is similar to the input sci_img and var_img.
- **gal_z** – Galaxy's redshift. This information will not be used in the image processing and only intended to be saved in the heder of a produced FITS file.
- **stamp_size** – Desired size for the reduced maps of multiband fluxes. This is a list data type with 2 elements. Accepted struture is: [dim_y,dim_x]. Only relevant if flag_crop=0.
- **remove_files** – If True, the unnecessary image files produced during the image processing will be removed. This can save disk space. If False, those files will not be removed.
- **idfil_align** – Index of the filter of which the image will be used as the reference in the spatial reprojection and sampling processes.

flux_map(*gal_region*, *output_stamps=None*, *Gal_EBV=None*, *scale_unit=1e-17*, *mag_zp_2mass=None*, *unit_spire='Jy_per_beam'*, *name_out_fits=None*)

Function for calculating maps of multiband fluxes from the stamp images produced by the *reduced_stamps()*.

Parameters

- **gal_region** – A 2D array containing the galaxy's region of interest. It is preferably the output of the `gal_region()`, but one can also make this input region. The 2D array should has the same size as that of the output stamps and the pixel value is 1 for the galaxy's region and 0 otherwise.
- **output_stamps** – (optional) Supply output_stamps dictionary input. This input is optional. If run_image_processing=False, this input is mandatory.
- **Gal_EBV** – The E(B-V) dust attenuation due to the foreground Galactic dust. This is optional parameter. If None, this value will be retrive from the IRSA data server through the astroquery package.

- **scale_unit** – Normalized unit for the fluxes in the output fits file. The unit is flux density in erg/s/cm^2/Ang.
- **mag_zp_2mass** – Magnitude zero-points of 2MASS images. Shoud be in 1D array with three elements: [magzp-j,magzp-h,magzp-k]. This is optional parameter. If not given (i.e. None), the values will be taken from the header of the FITS files.
- **unit_spire** – Unit of SPIRE images, in case Herschel/SPIRE image is included in the analysis. Therefore, this input is only relevant if Herschel/SPIRE image is among the images that are analyzed. Options are: ['Jy_per_beam', 'MJy_per_sr', 'Jy_per_pixel']
- **name_out_fits** – Desired name for the output FITS file. If None, a default name will be adopted.

galaxy_region(*segm_maps_ids=None*, *use_ellipse=False*, *x_cent=None*, *y_cent=None*, *ell=0*, *pa=45.0*, *radius_sma=30.0*)

Define galaxy's region of interest for further analysis.

Parameters

- **segm_maps_ids** – Array of IDs of selected segmentation maps (i.e., filters) to be used (merged) for constructing the galaxy's region. If None, all segmentation maps are merged together.
- **use_ellipse** – Alternative of defining galaxy's region using elliptical aperture centered at the target galaxy. Set use_ellipse=True if you want to use this option.
- **x_cent** – x coordinate of the ellipse center. If x_cent=None, the ellipse center is assumed to be the same as the image center.
- **y_cent** – y coordinate of the ellipse center. If y_cent=None, the ellipse center is assumed to be the same as the image center.
- **ell** – Ellipticity of the elliptical aperture.
- **pa** – Position angle of the elliptical aperture.
- **radius_sma** – Radal distance along the semi-major axis of the elliptical aperture. This radius is in pixel unit.

Returns gal_region
Output galaxy's region of interest.

get_output_stamps()

Get the names of output stamp images in a dictionary format.

plot_gal_region(*gal_region*, *output_stamps=None*, *ncols=6*, *savefig=True*, *name_plot=None*)

Plot the defined galaxy's region.

Parameters

- **gal_region** – The defined galaxy's region.
- **output_stamps** – (optional) Supply output_stamps dictionary input. This input is optional. If run_image_processing=False, this input is mandatory.
- **ncols** – Number of columns
- **savefig** – (default=True) Decide to save the plot or not.
- **name_plot** – (optional) Name for the output plot.

plot_image_stamps(*output_stamps=None*, *ncols=6*, *savefig=True*, *name_plot_sci=None*, *name_plot_var=None*)

Plotting resulted image stamps from the image processing.

Parameters

- **output_stamps** – (optional) Supply output_stamps dictionary input. This input is optional. If run_image_processing=False, this input is mandatory.
- **ncols** – Number of columns in the multipanel plots.
- **savefig** – (default=True) Decide to save the plot or not.
- **name_plot_sci** – (optional) Name for the output plot of science image stamps.
- **name_plot_var** – (optional) Name for the output plot of variance image stamps.

plot_segm_maps(*ncols=6*, *savefig=True*, *name_plot=None*)

Plotting segmentation maps.

Parameters

- **ncols** – Number of columns in the multipanel plots.
- **savefig** – (default=True) Decide to save the plot or not.
- **name_plot** – (optional) Name for the output plot.

rectangular_regions(*output_stamps=None*, *x=None*, *y=None*, *ra=None*, *dec=None*, *make_plot=True*, *ncols=6*, *savefig=True*, *name_plot=None*)

Define rectangular aperture and get pixels within it

Parameters

- **output_stamps** – (optional) Supply output_stamps dictionary input. This input is optional. If run_image_processing=False, this input is mandatory.
- **x** – x coordinates of the rectangular corners. The shape should be (n,4) with n is the number of rectangular apertures.
- **y** – y coordinates of the rectangular corners. The shape should be (n,4) with n is the number of rectangular apertures.
- **ra** – RA coordinates of the rectangular corners. The shape should be (n,4) with n is the number of rectangular apertures. If x input is given, this input will be ignored.
- **dec** – DEC coordinates of the rectangular corners. The shape should be (n,4) with n is the number of rectangular apertures. If y input is given, this inpur will be ignored.
- **pa** – Position angle of the rectangular aperture.
- **make_plot** – Decide to make a plot or not.
- **ncols** – Number of columns
- **savefig** – (default=True) Decide to save the plot or not.
- **name_plot** – (optional) Name for the output plot.

Returns rect_region

Rectangular aperture region.

reduced_stamps()

Run the image processing that includes PSF matching, spatial resampling and reprojection, and cropping around the target galaxy.

segmentation_sep(*output_stamps=None*, *thresh=1.5*, *minarea=30*, *deblend_nthresh=32*, *deblend_cont=0.005*)

Get segmentation maps of a galaxy in multiple bands using the SEP (a Python version of the SExtractor).

Parameters

- **output_stamps** – (optional) Supply output_stamps dictionary input. This input is optional. If run_image_processing=False, this input is mandatory.
- **thresh** – Detection threshold for the source detection and segmentation.
- **minarea** – Minimum number of pixels (above threshold) required for an object to be detected.
- **deblend_nthresh** – Number of deblending sub-thresholds. Default is 32.
- **deblend_cont** – Minimum contrast ratio used for object deblending. Default is 0.005. To entirely disable deblending, set to 1.0.

piXedfit.piXedfit_images.**kpc_per_pixel**(*z*, *arcsec_per_pix*, *cosmo='flat_LCDM'*, *H0=70.0*, *Om0=0.3*)

Function for calculating a physical scale (in kpc) corresponding to a single pixel in an image.

Parameters

- **z** – Redshift of the galaxy.
- **arsec_per_pix** – Pixel size in arcsecond.
- **cosmo** – Choices for the cosmology. The options are: (a)'flat_LCDM' or 0, (b)'WMAP5' or 1, (c)'WMAP7' or 2, (d)'WMAP9' or 3, (e)'Planck13' or 4, (f)'Planck15' or 5, (g)'Planck18' or 6. These are similar to the choices available in the Astropy Cosmology package.
- **H0** – The Hubble constant at z=0. Only relevant if cosmo='flat_LCDM'.
- **Om0** – The Omega matter at z=0.0. Only relevant if cosmo='flat_LCDM'.

Returns kpc_per_pix
Output physical scale in kpc.

piXedfit.piXedfit_images.**open_fluxmap_fits**(*flux_maps_fits*)

Function to extract data from the FITS file of fluxes maps produced from the image processing.

Parameters
flux_maps_fits – Input FITS file containing the flux maps.

Returns filters
The set of filters.

Returns gal_region
The galaxy's spatial region of interest.

Returns flux_map
The data cube of flux maps: (nbands,ny,nx).

Returns flux_err_map
The data cube of flux uncertainties maps: (nbands,ny,nx).

Returns unit_flux
The flux unit. In this case, the scale factor to erg/s/cm^2/Angstrom.

piXedfit.piXedfit_images.**photometry_within_aperture**(*flux_maps_fits*, *e=0.0*, *pa=45.0*, *cent_x=None*, *cent_y=None*, *radius=None*)

Function to get photometry within a given aperture (elliptical/circular) from input flux maps

Parameters

- `flux_maps_fits` – input FITS file containing the maps of fluxes produced from the image processing.
- `e` – Ellipticity of the apertures.
- `pa` – Position angle of the elliptical apertures.
- `cent_x` – The x coordinate of the central pixel.
- `cent_y` – The y coordinate of the central pixel.
- `radius` – The radius of the aperture.

Returns tot_fluxes
Output total fluxes.

Returns tot_flux_errors
Output total flux uncertainties.

`piXedfit.piXedfit_images.`**`plot_SED_pixels`**(*flux_maps_fits*, *pix_x=None*, *pix_y=None*, *all_pixels=False*, *logscale_y=True*, *logscale_x=True*, *wunit='angstrom'*, *yrange=None*, *xrange=None*, *savefig=True*, *name_plot=None*)

Function for plotting SEDs of pixels.

Parameters

- `flux_maps_fits` – Input FITS file of the multiband fluxes.
- `pix_x` – One dimensional array of x coordinates. Only relevant if all_pixels=False.
- `pix_y` – One dimensional array of the y coordinates. Only relevant if all_pixels=False.
- `all_pixels` – (optional) An option to get SEDs of all pixels.
- `logscale_y` – (optional) Option to set the y axis in logarithmic scale.
- `logscale_x` – (optional) Option to set the x axis in logarithmic scale.
- `wunit` – (optional) Wavelength unit. Options are 'angstrom' and 'micron'.
- `yrange` – (optional) Range in y axis.
- `xrange` – (optional) Range in x axis.
- `savefig` – (optional) Option to save the plot.
- `name_plot` – (optional) Name for the output plot.

`piXedfit.piXedfit_images.`**`plot_SNR_radial_profile`**(*flux_maps_fits*, *e=0.0*, *pa=45.0*, *cent_x=None*, *cent_y=None*, *yrange=None*, *xrange=None*, *savefig=True*, *name_plot=None*)

Function for plotting S/N ratios of pixels.

Parameters

- `flux_maps_fits` – Input FITS file produced from the image processing.
- `e` – Ellipticity of the elliptical apertures that will be used for deriving the S/N ratios.
- `pa` – The position angle of the elliptical apertures.
- `cent_x` – The x coordinate of the central pixel.
- `cent_y` – The y coordinate of the central pixel.
- `yrange` – Range in y-axis for the plot.

- **xrange** – Range in x-axis for the plot.
- **savefig** – Decide whether to save the plot or not.
- **name_plot** – Name for the output plot.

piXedfit.piXedfit_images.**plot_maps_fluxes**(*flux_maps_fits*, *ncols=5*, *savefig=True*, *name_plot_mapflux=None*, *name_plot_mapfluxerr=None*)

Function for plotting maps of multiband fluxes.

Parameters

- **flux_maps_fits** – Input FITS file of multiband flux maps.
- **ncols** – Number of columns in the plots.
- **savefig** – Decide whether to save the plot or not.
- **name_plot_mapflux** – Name of the output plot for the maps of multiband fluxes.
- **name_plot_mapfluxerr** – Name of the output plot for the maps of multiband flux uncertainties.

piXedfit.piXedfit_images.**radial_profile_psf**(*psf*, *pixsize*, *e=0.0*, *pa=45.0*, *dr_arcsec=None*)

Function to get the radial profile of PSF.

Parameters

- **psf** – Input PSF, either in a FITS file or the data array (2D).
- **e** – (default = 0) Ellipticity of the apertures. The default is zero (i.e., circular).
- **pa** – The position angle of the elliptical apertures.
- **dr_arsec** – Radial increment in units of arc second.

Pixsize
Pixel size in arc second.

Returns curve_rad
The radius array of the PSF radial profile.

Returns curve_val
The normalized fluxes of the PSF radial profile.

piXedfit.piXedfit_images.**subtract_background**(*fits_image*, *hdu_idx=0*, *sigma=3.0*, *box_size=None*, *mask_region=None*, *mask_sources=True*, *var_image=None*, *thresh=1.5*, *minarea=5*, *deblend_nthresh=32*, *deblend_cont=0.005*)

Function for estimating 2D background and subtracting it from the input image. This function also produce RMS image. This function adopts the Background2D function from the photutils package. To estimate 2D background, the input image is gridded and sigma clipping is done to each bin (grid). Then 2D interpolation is performed to construct the 2D background image. A more information can be seen at photutils website.

Parameters

- **fits_image** – Input image.
- **hdu_idx** – The extension (HDU) where the image is stored. Default is 0 (HDU0).
- **sigma** – Sigma clipping threshold value.
- **box_size** – The box size for the image gridding. The format is: [ny, nx]. If None, both axes will be divided into 10 grids.

- **mask_region** – Region within the image that are going to be excluded. mask_region should be 2D array with the same size as the input image.
- **mask_sources** – If True, source detection and segmentation will be performed with SEP (Pyhton version of SExtractor) and the regions associated with the detected sources will be excluded. This help reducing contamination by astronomical sources.
- **var_image** – Variance image (in FITS file format) to be used in the sources detection process. This input argument is only relevant if mask_sources=True.
- **thresh** – Detection threshold for the sources detection. If variance image is supplied, the threshold value for a given pixel is interpreted as a multiplicative factor of the uncertainty (i.e. square root of the variance) on that pixel. If var_image=None, the threshold is taken to be 2.5 percentile of the pixel values in the image.
- **minarea** – Minimum number of pixels above threshold triggering detection.
- **deblend_nthresh** – Number of thresholds used for object deblending.
- **deblend_cont** – Minimum contrast ratio used for object deblending. To entirely disable deblending, set to 1.0.

`piXedfit.piXedfit_images.`**`test_psfmatching_kernel`**(*init_PSF_name*, *target_PSF_name*, *kernel*, *pixscale_init_PSF*, *pixscale_target_PSF*, *dr_arcsec=None*, *xrange_arcsec=[-0.5, 0.5]*, *savefig=False*, *name_plot=None*)

Function for testing the convolution kernel. This includes convolving the initial PSF with the kernel, comparing the radial profiles of the convolved initial PSF and the target PSF along with the original initial PSF.

Parameters

- **init_PSF_name** – Image of the initial PSF.
- **target_PSF_name** – Image of the target PSF.
- **kernel** – The convolution kernel data (2D array).
- **pixscale_init_PSF** – Pixel size of the initial PSF in arcsec.
- **pixscale_target_PSF** – Pixel size of the target PSF in arcsec.
- **dr_arsec** – Radial increment for the radial profile in arcsec.
- **xrange_arcsec** – Range of x-axis in arsec.
- **savefig** – Decide whether to save the plot or not.
- **name_plot** – Name for the output plot.

`piXedfit.piXedfit_images.`**`var_img_2MASS`**(*sci_img*, *skyrms_img=None*, *skyrms_value=None*, *name_out_fits=None*)

Function for constructing a variance image from an input 2MASS image. The estimation of flux uncertainty follows the information provided on the 2MASS website.

Parameters

- **sci_img** – Input background-subtracted science image.
- **skyrms_img** – FITS file of the RMS background image. If skyrms_value=None, this parameter should be provided. The background subtraction and calculation of RMS image can be performed using the `subtract_background()` function.
- **skyrms_value** – Scalar value of median or mean of the RMS background image. This input will be considered when skyrms_img=None.

- **name_out_fits** – Desired name for the output FITS file.

piXedfit.piXedfit_images.**var_img_GALEX**(*sci_img*, *skybg_img*, *filter_name*, *name_out_fits=None*)

Function for calculating variance image from an input GALEX image

Parameters

- **sci_img** – Input GALEX science image (i.e., background subtracted). This type of image is provided in the GALEX website as indicated with "-intbgsub" (i.e., background subtracted intensity map).
- **skybg_img** – Input sky background image .
- **filter_name** – A string of filter name. Options are: 'galex_fuv' and 'galex_nuv'.
- **name_out_fits** – Desired name for the output variance image. If None, a generic name will be used.

piXedfit.piXedfit_images.**var_img_WISE**(*sci_img*, *unc_img*, *filter_name*, *skyrms_img*, *name_out_fits=None*)

Function for constructing variance image from an input WISE image. The uncertainty estimation follows the information from the WISE website

Parameters

- **sci_img** – Input background-subtracted science image.
- **unc_img** – Input uncertainty image. This type of WISE image is provided in the IRSA website and indicated with '-unc-' keyword.
- **filter_name** – The filter name. Options are: 'wise_w1', 'wise_w2', 'wise_w3', and 'wise_w4'
- **skyrms_img** – Input RMS background image. This image is produced in the background subtraction with the `subtract_background()` function.
- **name_out_fits** – (optional, default: None) Desired name for the output FITS file. If None, a generic name will be made.

piXedfit.piXedfit_images.**var_img_from_unc_img**(*unc_image*, *name_out_fits=None*)

Function for constructing a variance image from an input of uncertainty image. This function simply takes square of the uncertainty image and store it into a new FITS file while retaining the header information.

Parameters

- **unc_img** – Input uncertainty image.
- **name_out_fits** – (optional, default: None) Name of output FITS file. If None, a generic name will be generated.

piXedfit.piXedfit_images.**var_img_from_weight_img**(*wht_image*, *name_out_fits=None*)

Function for constructing a variance image from an input weight (i.e., inverse variance) image. This funciton simply takes inverse of the weight image and store it into a new FITS file while retaining the header information.

Parameters
wht_image – Input of weight image (i.e., inverse variance).

Returns name_out_fits
(optional, default: None) Name of output FITS file. If None, a generic name will be made.

piXedfit.piXedfit_images.**var_img_sdss**(*fits_image*, *filter_name*, *name_out_fits=None*)

A function for constructing a variance image from an input SDSS image

Parameters

- **fits_image** – Input SDSS image (corrected frame type).
- **filter_name** – A string of filter name. Options are: 'sdss_u', 'sdss_g', 'sdss_r', 'sdss_i', and 'sdss_z'.

Returns name_out_fits
Name of output FITS file. If None, a generic name will be used.

1.17 piXedfit_spectrophotometric

piXedfit.piXedfit_spectrophotometric.**match_imgifs_spatial**(*photo_fluxmap*, *ifs_data*, *ifs_survey='manga'*, *spec_smoothing=False*, *kernel_sigma=3.5*, *nproc=10*, *name_out_fits=None*)

Function for matching spatially (on pixel level) between IFS data cube and a post-processed multiwavelength imaging data cube. The current version of piXedfit only can process IFS data from CALIFA and MaNGA surveys.

Parameters

- **photo_fluxmap** – Input 3D data cube of photometry. This data cube is an output of function `flux_map()` in the `images_processing` class.
- **ifs_data** – Integral Field Spectroscopy (IFS) data cube.
- **ifs_survey** – The survey from which the IFS data cube is taken. Options are: 'manga' and 'califa'.
- **spec_smoothing** – If True, spectrum of each pixel will be smoothed by convolving it with a Gaussian kernel with a standard deviation given by the input kernel_sigma.
- **kernel_sigma** – Standard deviation of the kernel to be convolved with the spectrum of each pixel.
- **nproc** – Number of cores to be used for the calculation.
- **name_out_fits** – Desired name for the output FITS file.

piXedfit.piXedfit_spectrophotometric.**match_imgifs_spectral**(*specphoto_file*, *models_spec=None*, *nproc=10*, *del_wave_nebem=10.0*, *cosmo=0*, *H0=70.0*, *Om0=0.3*, *name_out_fits=None*)

Function for correcting wavelength-dependent mismatch between the IFS spectra and the photometric SEDs (on pixel level) in the spectrophotometric data cube that is produced using the function `piXedfit.piXedfit_spectrophotometric.match_imgifs_spatial()`.

Parameters

- **specphoto_file** – Input spectrophotometric data cube that is produced using `piXedfit.piXedfit_spectrophotometric.match_imgifs_spatial()`.
- **models_spec** – Set of model spectra at rest-frame produced using `piXedfit.piXedfit_model.save_models_rest_spec()`. The file is in the HDF5 format. For more information on how to produce this set of models, please see the description *here*. This set of models only need to be produced once and then it can be used for all galaxies in a sample. If models_spec=None, a default file is then called from piXedfit/data/mod ($PIXEDFIT_HOME/data/mod). However, this file is not available in that directory at first piXedfit is installed, instead user need to download it from this link and put it on that directory in the local machine.

- **nproc** – Number of cores to be used for calculation.
- **del_wave_nebem** – (default: 10.0 Angstrom). The range (+/-) around all emission lines in the model spectra that will be removed in producing spectral continuum, which will be used as reference for correcting the wavelength-dependent mismatch between the IFS spectra and photometric SEDs.
- **cosmo** – Choices for the cosmology. Options are: (a)'flat_LCDM' or 0, (b)'WMAP5' or 1, (c)'WMAP7' or 2, (d)'WMAP9' or 3, (e)'Planck13' or 4, (f)'Planck15' or 5. These options are the same to that available in the Astropy Cosmology package.
- **H0** – The Hubble constant at z=0. Only relevant when cosmo='flat_LCDM' is chosen.
- **Om0** – The Omega matter at z=0.0. Only relevant when cosmo='flat_LCDM' is chosen.
- **name_out_fits** – Desired name for the output FITS file. If None, a generic name will be used.

1.18 piXedfit_bin

piXedfit.piXedfit_bin.**get_bins_SED_binmap**(*binmap_fits*)

Function to extract SEDs of the spatial bins from the binned flux maps.

Parameters
binmap_fits – Input FITS file of binned flux maps.

Returns bin_photo_flux
Photometric fluxes of the spatial bins: (nbins,nbands)

Returns bin_photo_flux_err
Photometric flux uncertainties of the spatial bins: (nbins,nbands)

Returns bin_spec_flux
Spectroscopic fluxes of the spatial bins: (nbins,nwaves)

Returns bin_spec_flux_err
Spectroscopic flux uncertainties of the spatial bins: (nbins,nwaves)

Returns bin_flag_specphoto
A flag stating whether a spatial bin has a spectroscopy (value=1) or not (value=0).

Returns filters
The set of filters.

Returns photo_wave
The central wavelength of the filters.

Returns spec_wave
The wavelength grids of the spectra.

piXedfit.piXedfit_bin.**open_binmap_fits**(*binmap_fits*)

Function to get the data from binned flux maps.

Parameters
binmap_fits – Input FITS file containing binned flux maps.

Returns flag_specphoto
A flag stating whether the data contains spectra (value=1) or not (value=0).

Returns nbins_photo
Number of spatial bins with photometric data, which is also the total number of spatial bins.

Returns nbins_spec
Number of spatial bins that have spectra.

Returns filters
Set of the photometric filters.

Returns unit_flux
The flux unit.

Returns binmap_photo
The binning map of the photometric data.

Returns spec_region
The spatial region that have spectroscopy.

Returns binmap_spec
Number of spatial bins that have spectroscopy.

Returns map_photo_flux
The data cube of the photometric fluxes: (nbands,ny,nx)

Returns map_photo_flux_err
The data cube of the photometric flux uncertainties: (nbands,ny,nx)

Returns spec_wave
Spectroscopic wavelength grids.

Returns map_spec_flux
The data cube of the spectroscopic fluxes: (nwaves,ny,nx)

Returns map_spec_flux_err
The data cube of the spectroscopic flux uncertainties: (nwaves,ny,nx)

`piXedfit.piXedfit_bin.`**`pixel_binning`**(*fits_fluxmap*, *ref_band=None*, *Dmin_bin=4.0*, *SNR=None*, *redc_chi2_limit=4.0*, *del_r=2.0*, *name_out_fits=None*)

Function for pixel binning, a proses of combining neighboring pixels to optimize the signal-to-noise ratios of the spatially resolved SEDs. Input of this function is a data cube obtained from the image processing or spectrophotometric processing.

Parameters

- **`fits_fluxmap`** – Input FITS file containing the photometric or spectrophotometric data cube. The photometric data cube is obtained from the image processing with the `images_processing()` function, while the spectrophotmetric data cube is the output of function `match_imgifs_spectral()`.
- **`ref_band`** – Index of the reference band (filter) for sorting pixels based on the brightness. The central pixel of a bin is the brightest pixel in this reference band. If ref_band=None, the ref_band is chosen to be around the middle of the wavelength covered by the observed SEDs.
- **`Dmin_bin`** – Minimum diameter of a bin in unit of pixel.
- **`SNR`** – S/N thresholds in all bands. The length of this array should be the same as the number of bands in the fits_fluxmap. S/N threshold can vary across the filters. If SNR is None, the S/N is set as 5.0 to all the filters.
- **`redc_chi2_limit`** – A maximum reduced chi-square value for a pair of two SEDs to be considered as having a similar shape.
- **`del_r`** – Increment of circular radius (in unit of pixel) adopted in the pixel binning process.
- **`name_out_fits`** – Desired name for the output FITS file. If None, a default name is adopted.

piXedfit.piXedfit_bin.**pixel_binning_images**(*images*, *var_images*, *ref_band=None*, *Dmin_bin=2.0*, *SNR=None*, *redc_chi2_limit=4.0*, *del_r=2.0*, *name_out_fits=None*)

Function for pixel binning on multiband image.

Parameters

- **images** – Input science images. This input should be in a list format, such as images=['image1.fits', 'image2.fits', 'image3.fits']
- **var_images** – Variance images in a list format. The number of variance images should be the same as that of the input science images.
- **ref_band** – Index of the reference band (filter) for sorting pixels based on the brightness. The central pixel of a bin is the brightest pixel in this reference band. If ref_band=None, the ref_band is chosen to be around the middle of the wavelength covered by the observed SEDs.
- **Dmin_bin** – Minimum diameter of a bin in unit of pixel.
- **SNR** – S/N thresholds in all bands. The length of this array should be the same as the number of bands in the fits_fluxmap. S/N threshold can vary across the filters. If SNR is None, the S/N is set as 5.0 to all the filters.
- **redc_chi2_limit** – A maximum reduced chi-square value for a pair of two SEDs to be considered as having a similar shape.
- **del_r** – Increment of circular radius (in unit of pixel) adopted in the pixel binning process.
- **name_out_fits** – Desired name for the output FITS file. If None, a default name is adopted.

piXedfit.piXedfit_bin.**plot_binmap**(*binmap_fits*, *plot_binmap_spec=True*, *savefig=True*, *name_plot_binmap_photo=None*, *name_plot_binmap_spec=None*)

Function for plotting the binning map.

Parameters

- **binmap_fits** – Input FITS file of the binned flux maps.
- **plot_binmap_spec** – Decide to plot the binning map of pixels that have spectra. This input is only relevant if the data cube has both photometry and spectroscopy.
- **savefig** – Decide whether to save the plot of not.
- **name_plot_binmap_photo** – Name of the binning map of photometry.
- **name_plot_binmap_spec** – Name of the binning map of spectroscopy.

piXedfit.piXedfit_bin.**plot_bins_SED**(*binmap_fits*, *bin_ids=None*, *logscale_y=True*, *logscale_x=True*, *wunit='angstrom'*, *yrange=None*, *xrange=None*, *savefig=True*, *name_plot=None*)

Function for plotting SEDs of pixels.

Parameters

- **binmap_fits** – Input FITS file.
- **bin_ids** – Indexes of spatial bins which SEDs will be plotted. This index start from 0.
- **logscale_y** – (optional) Option to set the y axis in logarithmic scale.
- **logscale_x** – (optional) Option to set the x axis in logarithmic scale.
- **wunit** – (optional) Wavelength unit. Options are 'angstrom' and 'micron'.
- **yrange** – (optional) Range in y axis.

- **xrange** – (optional) Range in x axis.
- **savefig** – (optional) Option to save the plot.
- **name_plot** – (optional) Name for the output plot.

piXedfit.piXedfit_bin.**plot_bins_SNR_radial_profile**(*binmap_fits*, *xrange=None*, *yrange=None*, *savefig=True*, *name_plot=None*)

Function for plotting the S/N ratios of spatial bins.

Parameters

- **binmap_fits** – FITS file of binned flux maps produced by the pixel_binning function.
- **xrange** – Range in x-axis.
- **yrange** – Range in y-axis.
- **savefig** – Decide whether to save the plot or not.
- **name_plot** – Name of the output plot.

1.19 piXedfit_model

A description about the parameters involved in the SED modeling is given here.

piXedfit.piXedfit_model.**generate_modelSED_photo**(*filters*, *sp=None*, *imf_type=1*, *duste_switch=0*, *add_neb_emission=1*, *dust_law=1*, *sfh_form=4*, *add_agn=0*, *add_igm_absorption=0*, *igm_type=0*, *smooth_velocity=True*, *sigma_smooth=0.0*, *smooth_lsf=False*, *lsf_wave=None*, *lsf_sigma=None*, *cosmo='flat_LCDM'*, *H0=70.0*, *Om0=0.3*, *params_val={'dust1': 0.5, 'dust2': 0.5, 'dust_index': -0.7, 'gas_logu': -2.0, 'gas_logz': None, 'log_age': 1.0, 'log_alpha': 0.1, 'log_beta': 0.1, 'log_fagn': -3.0, 'log_gamma': -2.0, 'log_mass': 0.0, 'log_qpah': 0.54, 'log_t0': 0.4, 'log_tau': 0.4, 'log_tauagn': 1.0, 'log_umin': 0.0, 'logzsol': 0.0, 'z': 0.001}*)

Function for generating a model photometric SED given some parameters.

Parameters

- **filters** – List of photometric filters. The list of filters recognized by piXedfit can be accessed using `piXedfit.utils.filtering.list_filters()`. Please see this page for information on managing filters that include listing available filters, adding, and removing filters.
- **sp** – (optional, default: None) Initialization of FSPS, such as *sp=fsps.StellarPopulation()*. This is intended for rapid generation of model spectra from FSPS. However, this input is optional. If sp=None, FSPS will be called everytime this function is called.
- **imf_type** – Choice for the IMF. Choices are: 0 for Salpeter(1955), 1 for Chabrier(2003), and 2 for Kroupa(2001).
- **duste_switch** – Choice for switching on (value: 1) or off (value: 0) the dust emission modeling.
- **add_neb_emission** – Choice for switching on (value: 1) or off (value: 0) the nebular emission modeling.

- **dust_law** – Choice for the dust attenuation law. Options are: 0 for Charlot & Fall (2000) and 1 for Calzetti et al. (2000).
- **sfh_form** – Choice for the parametric SFH model. Options are: 0 for exponentially declining or tau model, 1 for delayed tau model, 2 for log normal model, 3 for Gaussian form, and 4 for double power-law model.
- **add_agn** – Choice for turning on (value: 1) or off (value: 0) the AGN dusty torus modeling.
- **add_igm_absorption** – Choice for turning on (value: 1) or off (value: 0) the IGM absorption modeling.
- **igm_type** – Choice for the IGM absorption model. Options are: 0 for Madau (1995) and 1 for Inoue+(2014).
- **smooth_velocity** – (default: True) The same parameter as in FSPS. Switch to perform smoothing in velocity space (if True) or wavelength space.
- **sigma_smooth** – (default: 0.0) The same parameter as in FSPS. If smooth_velocity is True, this gives the velocity dispersion in km/s. Otherwise, it gives the width of the gaussian wavelength smoothing in Angstroms. These widths are in terms of sigma (standard deviation), not FWHM.
- **smooth_lsf** – (default: False) The same parameter as in FSPS. Switch to apply smoothing of the SSPs by a wavelength dependent line spread function. Only takes effect if smooth_velocity is True.
- **lsf_wave** – Wavelength grids for the input line spread function. This must be in the units of Angstroms, and sorted ascending.
- **lsf_sigma** – The dispersion of the Gaussian line spread function at the wavelengths given by lsf_wave, in km/s. This array must have the same length as lsf_wave. If value is 0, no smoothing will be applied at that wavelength.
- **cosmo** – Choices for the cosmology. Options are: (1)'flat_LCDM' or 0, (2)'WMAP5' or 1, (3)'WMAP7' or 2, (4)'WMAP9' or 3, (5)'Planck13' or 4, (6)'Planck15' or 5. These options are similar to the choices available in the Astropy Cosmology package.
- **H0** – The Hubble constant at z=0. Only relevant when cosmo='flat_LCDM' is chosen.
- **Om0** – The Omega matter at z=0.0. Only relevant when cosmo='flat_LCDM' is chosen.
- **param_val** – Dictionary of the input values of the parameters. Should folllow the structure given in the default set. Summary of the parameters are tabulated in Table 1 of Abdurro'uf et al. (2021).

Returns photo_SED

Output model photometric SED. It consists of photo_SED['wave'], which is the central wavelengths of the photometric filters, and photo_SED['flux'], which is the photometric fluxes.

`piXedfit.piXedfit_model.`**`generate_modelSED_spec`**(*sp=None*, *imf_type=1*, *duste_switch=1*, *add_neb_emission=1*, *dust_law=1*, *sfh_form=4*, *add_agn=0*, *add_igm_absorption=0*, *igm_type=0*, *smooth_velocity=True*, *sigma_smooth=0.0*, *smooth_lsf=False*, *lsf_wave=None*, *lsf_sigma=None*, *cosmo='flat_LCDM'*, *H0=70.0*, *Om0=0.3*, *params_val={'dust1': 0.5, 'dust2': 0.5, 'dust_index': -0.7, 'gas_logu': -2.0, 'gas_logz': None, 'log_age': 1.0, 'log_alpha': 0.1, 'log_beta': 0.1, 'log_fagn': -3.0, 'log_gamma': -2.0, 'log_mass': 0.0, 'log_qpah': 0.54, 'log_t0': 0.4, 'log_tau': 0.4, 'log_tauagn': 1.0, 'log_umin': 0.0, 'logzsol': 0.0, 'z': 0.001}*, *add_neb_continuum=1*)

Function for generating a model spectrum given some parameters.

Parameters

- **sp** – (optional, default: None) Initialization of FSPS, such as *sp=fsps.StellarPopulation()*. This is intended for rapid generation of model spectra from FSPS. However, this input is optional. If sp=None, FSPS will be called everytime this function is called.
- **imf_type** – Choice for the IMF. Choices are: 0 for Salpeter(1955), 1 for Chabrier(2003), and 2 for Kroupa(2001).
- **duste_switch** – Choice for switching on (value: 1) or off (value: 0) the dust emission modeling.
- **add_neb_emission** – Choice for switching on (value: 1) or off (value: 0) the nebular emission modeling.
- **dust_law** – Choice for the dust attenuation law. Options are: 0 for Charlot & Fall (2000) and 1 for Calzetti et al. (2000).
- **sfh_form** – Choice for the parametric SFH model. Options are: 0 for exponentially declining or tau model, 1 for delayed tau model, 2 for log normal model, 3 for Gaussian form, and 4 for double power-law model.
- **add_agn** – Choice for turning on (value: 1) or off (value: 0) the AGN dusty torus modeling.
- **add_igm_absorption** – Choice for turning on (value: 1) or off (value: 0) the IGM absorption modeling.
- **igm_type** – Choice for the IGM absorption model. Options are: 0 for Madau (1995) and 1 for Inoue+(2014).
- **smooth_velocity** – (default: True) The same parameter as in FSPS. Switch to perform smoothing in velocity space (if True) or wavelength space.
- **sigma_smooth** – (default: 0.0) The same parameter as in FSPS. If smooth_velocity is True, this gives the velocity dispersion in km/s. Otherwise, it gives the width of the gaussian wavelength smoothing in Angstroms. These widths are in terms of sigma (standard deviation), not FWHM.
- **smooth_lsf** – (default: False) The same parameter as in FSPS. Switch to apply smoothing of the SSPs by a wavelength dependent line spread function. Only takes effect if smooth_velocity is True.
- **lsf_wave** – Wavelength grids for the input line spread function. This must be in the units of Angstroms, and sorted ascending.

- **lsf_sigma** – The dispersion of the Gaussian line spread function at the wavelengths given by lsf_wave, in km/s. This array must have the same length as lsf_wave. If value is 0, no smoothing will be applied at that wavelength.
- **cosmo** – Choices for the cosmology. Options are: (1)'flat_LCDM' or 0, (2)'WMAP5' or 1, (3)'WMAP7' or 2, (4)'WMAP9' or 3, (5)'Planck13' or 4, (6)'Planck15' or 5. These options are similar to the choices available in the Astropy Cosmology package.
- **H0** – The Hubble constant at z=0. Only relevant when cosmo='flat_LCDM' is chosen.
- **Om0** – The Omega matter at z=0.0. Only relevant when cosmo='flat_LCDM' is chosen.
- **param_val** – Dictionary of the input values of the parameters. Should folllow the structure given in the default set. Summary of the parameters are tabulated in Table 1 of Abdurro'uf et al. (2021).

Returns spec_SED
Array containing output model spectrum. It consists of spec_SED['wave'], which is the wavelengths grids, and spec_SED['flux'], which is the fluxes or the spectrum.

piXedfit.piXedfit_model.**save_models_photo**(*filters*, *gal_z*, *imf_type=1*, *sfh_form=4*, *dust_law=1*, *add_igm_absorption=0*, *igm_type=0*, *duste_switch=0*, *add_neb_emission=1*, *add_agn=0*, *nmodels=100000*, *nproc=10*, *smooth_velocity=True*, *sigma_smooth=0.0*, *smooth_lsf=False*, *lsf_wave=None*, *lsf_sigma=None*, *cosmo='flat_LCDM'*, *H0=70.0*, *Om0=0.3*, *params_range={'dust1': [0.0, 4.0], 'dust2': [0.0, 4.0], 'dust_index': [-2.2, 0.4], 'gas_logu': [-4.0, -1.0], 'gas_logz': None, 'log_age': [-2.0, 1.14], 'log_alpha': [-2.0, 2.0], 'log_beta': [-2.0, 2.0], 'log_fagn': [-5.0, 0.48], 'log_gamma': [-4.0, 0.0], 'log_qpah': [-3.0, 1.0], 'log_t0': [-1.0, 1.14], 'log_tau': [-1.0, 1.5], 'log_tauagn': [0.7, 2.18], 'log_umin': [-1.0, 1.39], 'logzsol': [-2.0, 0.2]}*, *name_out=None*)

Function for generating a set of photometric model SEDs and store them into a FITS file. The values of the parameters are randomly generated and for each parameter, the random values are uniformly distributed.

Parameters

- **filters** – List of photometric filters. The list of filters recognized by piXedfit can be accessed using `piXedfit.utils.filtering.list_filters()`. Please see this page for information on managing filters that include listing available filters, adding, and removing filters.
- **gal_z** – Galaxy's redshift.
- **imf_type** – Choice for the IMF. Choices are: 0 for Salpeter(1955), 1 for Chabrier(2003), and 2 for Kroupa(2001).
- **sfh_form** – Choice for the parametric SFH model. Options are: 0 for exponentially declining or tau model, 1 for delayed tau model, 2 for log normal model, 3 for Gaussian form, and 4 for double power-law model.
- **dust_law** – Choice for the dust attenuation law. Options are: 0 for Charlot & Fall (2000) and 1 for Calzetti et al. (2000).
- **add_igm_absorption** – Choice for turning on (value: 1) or off (value: 0) the IGM absorption modeling.
- **igm_type** – Choice for the IGM absorption model. Options are: 0 for Madau (1995) and 1 for Inoue+(2014).

- **duste_switch** – Choice for switching on (value: 1) or off (value: 0) the dust emission modeling.
- **add_neb_emission** – Choice for switching on (value: 1) or off (value: 0) the nebular emission modeling.
- **add_agn** – Choice for turning on (value: 1) or off (value: 0) the AGN dusty torus modeling.
- **nmodels** – Number of model SEDs to be generated.
- **params_range** – Ranges of parameters in a dictionary format. Summary of the parameters are tabulated in Table 1 of Abdurro'uf et al. (2021).
- **nproc** – Number of cores to be used in the calculations.
- **smooth_velocity** – (default: True) The same parameter as in FSPS. Switch to perform smoothing in velocity space (if True) or wavelength space.
- **sigma_smooth** – (default: 0.0) The same parameter as in FSPS. If smooth_velocity is True, this gives the velocity dispersion in km/s. Otherwise, it gives the width of the gaussian wavelength smoothing in Angstroms. These widths are in terms of sigma (standard deviation), not FWHM.
- **smooth_lsf** – (default: False) The same parameter as in FSPS. Switch to apply smoothing of the SSPs by a wavelength dependent line spread function. Only takes effect if smooth_velocity is True.
- **lsf_wave** – Wavelength grids for the input line spread function. This must be in the units of Angstroms, and sorted ascending.
- **lsf_sigma** – The dispersion of the Gaussian line spread function at the wavelengths given by lsf_wave, in km/s. This array must have the same length as lsf_wave. If value is 0, no smoothing will be applied at that wavelength.
- **cosmo** – Choices for the cosmology. Options are: (1)'flat_LCDM' or 0, (2)'WMAP5' or 1, (3)'WMAP7' or 2, (4)'WMAP9' or 3, (5)'Planck13' or 4, (6)'Planck15' or 5. These options are similar to the choices available in the Astropy Cosmology package.
- **H0** – The Hubble constant at z=0. Only relevant when cosmo='flat_LCDM' is chosen.
- **Om0** – The Omega matter at z=0.0. Only relevant when cosmo='flat_LCDM' is chosen.

Returns name_out_fits

Desired name for the output FITS file. if None, a default name will be used.

`piXedfit.piXedfit_model.`**`save_models_rest_spec`**(*imf_type=1*, *sfh_form=4*, *dust_law=1*, *duste_switch=0*, *add_neb_emission=1*, *add_agn=0*, *nmodels=100000*, *nproc=5*, *smooth_velocity=True*, *sigma_smooth=0.0*, *smooth_lsf=False*, *lsf_wave=None*, *lsf_sigma=None*, *params_range={'dust1': [0.0, 4.0], 'dust2': [0.0, 4.0], 'dust_index': [-2.2, 0.4], 'gas_logu': [-4.0, -1.0], 'gas_logz': None, 'log_age': [-2.0, 1.14], 'log_alpha': [-2.0, 2.0], 'log_beta': [-2.0, 2.0], 'log_fagn': [-5.0, 0.48], 'log_gamma': [-4.0, 0.0], 'log_qpah': [-3.0, 1.0], 'log_t0': [-1.0, 1.14], 'log_tau': [-1.0, 1.5], 'log_tauagn': [0.7, 2.18], 'log_umin': [-1.0, 1.39], 'logzsol': [-2.0, 0.2]}*, *name_out=None*)

Function for generating a set of model spectra at rest-frame. The values of the parameters are randomly generated and for each parameter, the random values are uniformly distributed.

Parameters

- **imf_type** – Choice for the IMF. Choices are: 0 for Salpeter(1955), 1 for Chabrier(2003), and 2 for Kroupa(2001).
- **sfh_form** – Choice for the parametric SFH model. Options are: 0 for exponentially declining or tau model, 1 for delayed tau model, 2 for log normal model, 3 for Gaussian form, and 4 for double power-law model.
- **dust_law** – Choice for the dust attenuation law. Options are: 0 for Charlot & Fall (2000) and 1 for Calzetti et al. (2000).
- **duste_switch** – Choice for switching on (value: 1) or off (value: 0) the dust emission modeling.
- **add_neb_emission** – Choice for switching on (value: 1) or off (value: 0) the nebular emission modeling.
- **add_agn** – Choice for turning on (value: 1) or off (value: 0) the AGN dusty torus modeling.
- **nmodels** – Number of model SEDs to be generated.
- **params_range** – Ranges of parameters in a dictionary format. Summary of the parameters are tabulated in Table 1 of Abdurro'uf et al. (2021).
- **nproc** – Number of cores to be used in the calculations.
- **smooth_velocity** – (default: True) The same parameter as in FSPS. Switch to perform smoothing in velocity space (if True) or wavelength space.
- **sigma_smooth** – (default: 0.0) The same parameter as in FSPS. If smooth_velocity is True, this gives the velocity dispersion in km/s. Otherwise, it gives the width of the gaussian wavelength smoothing in Angstroms. These widths are in terms of sigma (standard deviation), not FWHM.
- **smooth_lsf** – (default: False) The same parameter as in FSPS. Switch to apply smoothing of the SSPs by a wavelength dependent line spread function. Only takes effect if smooth_velocity is True.
- **lsf_wave** – Wavelength grids for the input line spread function. This must be in the units of Angstroms, and sorted ascending.
- **lsf_sigma** – The dispersion of the Gaussian line spread function at the wavelengths given by lsf_wave, in km/s. This array must have the same length as lsf_wave. If value is 0, no smoothing will be applied at that wavelength.

Returns name_out
Name for the output HDF5 file.

1.20 piXedfit_fitting

`piXedfit.piXedfit_fitting.SEDfit_from_binmap(fits_binmap, binid_range=None, bin_ids=None, models_spec=None, params_ranges=None, params_priors=None, fit_method='mcmc', gal_z=None, free_z=0, nrands_z=10, wavelength_range=None, smooth_velocity=True, sigma_smooth=0.0, spec_resolution=None, smooth_lsf=False, lsf_wave=None, lsf_sigma=None, poly_order=10, spec_chi_sigma_clip=4.0, del_wave_nebem=15.0, add_igm_absorption=0, igm_type=0, likelihood='gauss', dof=3.0, nwalkers=100, nsteps=600, nsteps_cut=50, nproc=10, initfit_nmodels_mcmc=100000, perc_chi2=90.0, cosmo=0, H0=70.0, Om0=0.3, store_full_samplers=1, name_out_fits=None)`

Function for performing SED fitting to a photometric data cube. The data cube should has been binned using the function `pixel_binning()`.

Parameters

- **fits_binmap** – Input FITS file of the spectrophotometric data cube that has been binned.
- **binid_range** – Range of bin IDs which the SEDs are going to be fit. The accepted format is [idmin,idmax]. The ID starts from 0. If None, the SED fitting will be done to all of the spatial bins in the galaxy.
- **bin_ids** – Bin IDs which the SEDs are going to be fit. The accepted format is 1D array. The ID starts from 0. Both binid_range and bin_ids can't be None. If both of them are not None, the bin_ids will be used.
- **models_spec** – Model spectral templates in the rest-frame generated prior to the SED fitting process using the function *piXedfit.piXedfit_model.save_models_rest_spec()*. This set of model spectra will be used in the main fitting step if fit_method='rdsps' or initial fitting if fit_method='mcmc'.
- **params_ranges** – Ranges of the parameter defined (i.e., outputted) using the `params_ranges()` in the *priors* class.
- **params_priors** – Forms of adopted priros. The acceptable format is a list, such as params_priors=[prior1, prior2, prior3] where prior1, prior2, and prior3 are output of functions in the *priors* class.
- **fit_method** – (default: 'mcmc') Choice for the fitting method. Options are: 'mcmc' and 'rdsps'.
- **gal_z** – Redshift of the galaxy. If gal_z=None, then redshift is taken from the header of the FITS file. If gal_z in the FITS header is negative, the redshift is set to be free in the SED fitting (i.e., photometric redshift).
- **free_z** – A flag stating whether redshift would be free prameter (value: 1) or not (value: 0). If free_z=1, the gal_z input is not relevant, but the redshift range that is set when setting priors would be considered.
- **nrands_z** – Number of random redshifts to be generated (within the chosen range as set in the params_range) in the main fitting if fit_method='rdsps' or initial fitting if fit_method='mcmc'. This is only relevant if gal_z=None (i.e., photometric redshift will be activated).
- **wavelength_range** – Range of wavelength within which the observed spectrum will be considered in the SED fitting. The accepted format is [wmin,wmax] with wmin and wmax are minimum and maximum wavelengths.

- **smooth_velocity** – (default: True) The same parameter as in FSPS. Switch to perform smoothing in velocity space (if True) or wavelength space.
- **sigma_smooth** – (default: 0.0) The same parameter as in FSPS. If smooth_velocity is True, this gives the velocity dispersion in km/s. Otherwise, it gives the width of the gaussian wavelength smoothing in Angstroms. These widths are in terms of sigma (standard deviation), not FWHM.
- **spec_resolution** – (default: None) Spectral resolution (R) of the input spectra. This is R=c/sigma_smooth if sigma_smooth is a velocity dispersion. This parameter will be considered if smooth_velocity=True and sigma_smooth=None. The sigma_smooth will then be calculated using the above equation.
- **smooth_lsf** – (default: False) The same parameter as in FSPS. Switch to apply smoothing of the SSPs by a wavelength dependent line spread function. Only takes effect if smooth_velocity is True.
- **lsf_wave** – Wavelength grids for the input line spread function. This must be in the units of Angstroms, and sorted ascending.
- **lsf_sigma** – The dispersion of the Gaussian line spread function at the wavelengths given by lsf_wave, in km/s. This array must have the same length as lsf_wave. If value is 0, no smoothing will be applied at that wavelength.
- **poly_order** – The degree of the legendre polynomial function to be used for correcting the shape (normalization) of the model spectra.
- **spec_chi_sigma_clip** – Standard deviation (sigma) to be adopted in the sigma clipping to the spectrum data points that are regarded as outliers before calculating chi-square in the SED fitting process. The sigma clipping is carried out based on the distribution of chi values (sum((D-M)/Derr)).
- **del_wave_nebem** – This parameter defines the Wavelength region (+/- del_wave_nebem) around the emission lines that will be excluded in the fitting of spectral continuum between the model spectrum and the observed one.
- **add_igm_absorption** – Switch for the IGM absorption. Options are: 0 for switch off and 1 for switch on.
- **igm_type** – Choice for the IGM absorption model. Options are: 0 for Madau (1995) and 1 for Inoue+(2014).
- **likelihood** – Choice of likelihood function for the RDSPS method. Only relevant if the fit_method='rdsps'. Options are: 'gauss' for the Gaussian form and 'student_t' for the student's t form.
- **dof** – Degree of freedom (nu) in the Student's t likelihood function. Only relevant if the fit_method='rdsps' and likelihood='student_t'.
- **nwalkers** – Number of walkers in the MCMC process. This parameter is only applicable if fit_method='mcmc'.
- **nsteps** – Number of steps for each walker in the MCMC process. Only relevant if fit_method='mcmc'.
- **nsteps_cut** – Number of first steps of each walkers that will be cut when collecting the final sampler chains. Only relevant if fit_method='mcmc' and store_full_samplers=1.
- **nproc** – Number of processors (cores) to be used in the calculation.
- **initfit_nmodels_mcmc** – Number of models to be used in the initial fitting in the MCMC method. Only relevant if fit_method='mcmc'.

- **perc_chi2** – A percentile in the set of models sorted based on the chi-square values that will be considered in the calculation of the best-fit parameters (i.e., posterior-weighted averages) in the RDSPS fitting. This parameter is only relevant if fit_method='rdsps'.
- **cosmo** – Choices for the cosmology. Options are: (a)'flat_LCDM' or 0, (b)'WMAP5' or 1, (c)'WMAP7' or 2, (d)'WMAP9' or 3, (e)'Planck13' or 4, (f)'Planck15' or 5. These options are similar to the choices available in the Astropy Cosmology package.
- **H0** – The Hubble constant at z=0. Only relevant when cosmo='flat_LCDM' is chosen.
- **Om0** – The Omega matter at z=0.0. Only relevant when cosmo='flat_LCDM' is chosen.
- **store_full_samplers** – Flag indicating whether full sampler models will be stored into the output FITS file or not. Options are: 1 or True for storing the full samplers and 0 or False otherwise.
- **name_out_fits** – Names of output FITS files. This parameter is optional. If not None it must be in a list format with the same number of elements as the number of bins to be performed SED fitting. Example: name_out_fits = ['bin1.fits', 'bin2.fits', . . . , 'binN.fits']. If None, default names will be adopted.

`piXedfit.piXedfit_fitting.`**`get_bestfit_params`**`(`*input_fits*`)`

Function to get (i.e., read) best-fit parameters from the FITS file output of the fitting process.

Parameters
input_fits – Input FITS file, which is an output of the fitting process.

Returns params
The list of parameters.

Returns bfit_params
A dictionary containing the best-fit parameters.

`piXedfit.piXedfit_fitting.`**`maps_parameters`**`(`*fits_binmap*, *bin_ids*, *name_sampler_fits*, *fits_fluxmap=None*, *refband_SFR=None*, *refband_SM=None*, *refband_Mdust=None*, *name_out_fits=None*`)`

Function for constructing maps of properties of a galaxy from the collectin of fitting results of the spatial bins within the galaxy.

Parameters

- **fits_binmap** – Input FITS file of the spectrophotometric data cube that has been binned.
- **bin_ids** – Bin indices of the FITS files listed in name_sampler_fits input. Allowed format is a 1D array. The id starts from 0.
- **name_sampler_fits** – List of the names of the FITS files containing the fitting results of spatial bins. This should have the same number of element as that of bin_ids. The number of element doesn't necessarily the same as the number of bins. A missing bin will be ignored in the creation of the maps of properties.
- **fits_fluxmap** – FITS file containing reduced maps of multiband fluxes, which is output of the `flux_map()` in the `images_processing` class in the `piXedfit_images` module.
- **refband_SFR** – Index of band in the multiband set that is used for reference in dividing map of SFR in bin space into map of SFR in pixel space. If None, the band with shortest wavelength is selected.
- **refband_SM** – Index of band in the multiband set that is used for reference in dividing map of stellar mass in bin space into map of stellar mass in pixel space. If None, the band with longest wavelength is selected.

- **refband_Mdust** – Index of band/filter in the multiband set that is used for reference in dividing map of dust mass in bin space into map of dust mass in pixel space. If None, the band with longest wavelength is selected.
- **name_out_fits** – Desired name for the output FITS file. If None, a default name will be used.

`class piXedfit.piXedfit_fitting.`**`priors`**(*ranges={'dust1': [0.0, 4.0], 'dust2': [0.0, 4.0], 'dust_index': [-2.2, 0.4], 'gas_logu': [-4.0, -1.0], 'gas_logz': None, 'log_age': [-1.0, 1.14], 'log_alpha': [-2.0, 2.0], 'log_beta': [-2.0, 2.0], 'log_fagn': [-5.0, 0.48], 'log_gamma': [-4.0, 0.0], 'log_mass': [4.0, 12.0], 'log_mw_age': [-2.0, 1.14], 'log_qpah': [-3.0, 1.0], 'log_t0': [-1.0, 1.14], 'log_tau': [-1.0, 1.5], 'log_tauagn': [0.7, 2.18], 'log_umin': [-1.0, 1.39], 'logzsol': [-2.0, 0.2], 'z': [0.0, 1.0]}*)

Functions for defining priors to be used in the Bayesian SED fitting process. First, one need to define ranges for the parameters using `params_ranges()`, then define shape of the prior distribution function of each parameter. The available analytic forms for the prior are uniform, Gaussian, Student's t, and gamma functions. User can also choose an arbitrary one, using `arbitrary()`. It is also possible to define a joint prior between a given parameter and stellar mass. This joint prior can be adopted from a known scaling relation, such as stellar mass vs metallicity relation. Note that it is not required to define range of all parameters to be involved in SED fitting. If range is not inputted, the default one will be used. It is also not required to define prior shape of all parameters. If not inputted, a uniform prior will be used. Due to the defined ranges, the analytical form is truncated at the edges defined by the range.

Parameters

ranges – The ranges of the parameters. If *gas_logz* is None, gas-phase metallicity is set to have the same value as the stellar metallicity.

`arbitrary`(*param*, *values*, *prob*)

Function for assigning an arbitrary prior.

Parameters

- **param** – The parameter to be assigned with the arbitrary prior.
- **values** – Array of values.
- **prob** – Array of probability associated with the values.

Returns prior

Output prior.

`gamma`(*param*, *a*, *loc*, *scale*)

Function for assigning a prior in the form of Gamma function to a parameter.

Parameters

- **param** – The parameter that will be assigned the Gamma prior.
- **a** – A shape parameter in the gamma function.
- **loc** – Peak location.
- **scale** – Width of the distribution.

Returns prior

Output prior.

`gaussian`(*param*, *loc*, *scale*)

Function for assigning Gaussian prior to a parameter.

Parameters

- **param** – The parameter that will be assigned the Gaussian prior.
- **loc** – Peak location of the Gaussian prior.
- **scale** – Width or standard deviation of the Gaussian prior.

Returns prior
Output prior.

joint_with_mass(*param*, *log_mass*, *param_values*, *scale*)

Function for assigning a joint prior between a given parameter and stellar mass (log_mass).

Parameters

- **param** – The parameter that will share a joint prior with the stellar mass.
- **log_mass** – Array of stellar mass values.
- **param_values** – Array of the parameter values. In this case, the parameter that shares a joint prior with the stellar mass.
- **scale** – Array of width or standard deviations of the param_value.

Returns prior
Output prior.

params_ranges()

Function for defining ranges of the parameters.

Returns params_ranges
Ranges of the parameters to be inputted into a SED fitting function.

studentt(*param*, *df*, *loc*, *scale*)

Function for assigning a prior in the form of Student's t distribution.

Parameters

- **param** – The parameter that will be assigned the Student's t prior.
- **df** – The degree of freedom.
- **loc** – Peak location.
- **scale** – Width of the distribution.

Returns prior
Output prior.

uniform(*param*)

Function for assigning uniform prior to a parameter.

Parameters
param – The parameter that will be assigned the uniform prior.

Returns prior
Output prior.

`piXedfit.piXedfit_fitting.singleSEDfit`(*obs_flux=None*, *obs_flux_err=None*, *filters=None*, *spec_wave=None*, *spec_flux=None*, *spec_flux_err=None*, *gal_z=None*, *models_spec=None*, *wavelength_range=None*, *params_ranges=None*, *params_priors=None*, *fit_method='mcmc'*, *nrands_z=10*, *add_igm_absorption=0*, *igm_type=0*, *smooth_velocity=True*, *sigma_smooth=0.0*, *spec_resolution=None*, *smooth_lsf=False*, *lsf_wave=None*, *lsf_sigma=None*, *poly_order=10*, *spec_chi_sigma_clip=5.0*, *del_wave_nebem=15.0*, *likelihood='gauss'*, *dof=2.0*, *nwalkers=100*, *nsteps=600*, *nsteps_cut=50*, *nproc=10*, *initfit_nmodels_mcmc=100000*, *perc_chi2=90.0*, *cosmo=0*, *H0=70.0*, *Om0=0.3*, *store_full_samplers=1*, *name_out_fits=None*)

Function for performing SED fitting to a single SED. The input SED can be in three forms: (1) photometry only (with input obs_flux, obs_flux_err, filters and leave spec_wave=None,spec_flux=None,spec_flux_err=None), (2) spectrum only (with input spec_wave, spec_flux, and spec_flux_err while leaving obs_flux=None,obs_flux_err=None,filters=None), and (3) spectrophotometry if all inputs are provided.

Parameters

- **obs_flux** – Input fluxes in multiple bands. The format is 1D array with a number of elements of the array should be the sama as that of obs_flux_err and filters. The fluxes should be in the unit of erg/s/cm^2/Angstrom.
- **obs_flux_err** – Input flux uncertainties in multiple bands. The flux uncertainties should be in the unit of erg/s/cm^2/Angstrom.
- **filters** – List of photometric filters. The list of filters recognized by piXedfit can be accesses using *piXedfit.utils.filtering.list_filters()*. Please see this page for information on managing filters that include listing available filters, adding, and removing filters.
- **spec_wave** – 1D array of wavelength of the input spectrum.
- **spec_flux** – Flux grids of the input spectrum. The fluxes should be in the unit of erg/s/cm^2/Angstrom.
- **spec_flux_err** – Flux uncertainties of the input spectrum. The flux uncertainties should be in the unit of erg/s/cm^2/Angstrom.
- **gal_z** – Redshift of the galaxy. If gal_z=None, redshift will be a free parameter in the fitting.
- **models_spec** – Model spectral templates in the rest-frame generated prior to the SED fitting process using the function *piXedfit.piXedfit_model.save_models_rest_spec()*. This set of model spectra will be used in the main fitting step if fit_method='rdsps' or initial fitting if fit_method='mcmc'.
- **wavelength_range** – Range of wavelength within which the observed spectrum will be considered in the SED fitting. The accepted format is [wmin,wmax] with wmin and wmax are minimum and maximum wavelengths.
- **params_ranges** – Ranges of the parameter defined (i.e., outputted) using the `params_ranges()` in the *priors* class.
- **params_priors** – Forms of adopted priros. The acceptable format is a list, such as params_priors=[prior1, prior2, prior3] where prior1, prior2, and prior3 are output of functions in the *priors* class.
- **fit_method** – Choice of method for the SED fitting. Options are: (a)'mcmc' for Markov Chain Monte Carlo, and (b)'rdsps' for Random Dense Sampling of Parameter Space.

- **nrands_z** – Number of random redshifts to be generated (within the chosen range as set in the params_range) in the main fitting if fit_method='rdsps' or initial fitting if fit_method='mcmc'. This is only relevant if gal_z=None (i.e., photometric redshift will be activated).
- **add_igm_absorption** – Switch for the IGM absorption. Options are: 0 for switch off and 1 for switch on.
- **igm_type** – Choice for the IGM absorption model. Options are: 0 for Madau (1995) and 1 for Inoue+(2014).
- **smooth_velocity** – (default: True) The same parameter as in FSPS. Switch to perform smoothing in velocity space (if True) or wavelength space.
- **sigma_smooth** – (default: 0.0) The same parameter as in FSPS. If smooth_velocity is True, this gives the velocity dispersion in km/s. Otherwise, it gives the width of the gaussian wavelength smoothing in Angstroms. These widths are in terms of sigma (standard deviation), not FWHM.
- **spec_resolution** – (default: None) Spectral resolution (R) of the input spectra. This is R=c/sigma_smooth if sigma_smooth is a velocity dispersion. This parameter will be considered if smooth_velocity=True and sigma_smooth=None. The sigma_smooth will then be calculated using the above equation.
- **smooth_lsf** – (default: False) The same parameter as in FSPS. Switch to apply smoothing of the SSPs by a wavelength dependent line spread function. Only takes effect if smooth_velocity is True.
- **lsf_wave** – Wavelength grids for the input line spread function. This must be in the units of Angstroms, and sorted ascending.
- **lsf_sigma** – The dispersion of the Gaussian line spread function at the wavelengths given by lsf_wave, in km/s. This array must have the same length as lsf_wave. If value is 0, no smoothing will be applied at that wavelength.
- **poly_order** – The degree of the legendre polynomial function to be used for correcting the shape (normalization) of the model spectra.
- **spec_chi_sigma_clip** – Standard deviation (sigma) to be adopted in the sigma clipping to the spectrum data points that are regarded as outliers before calculating chi-square in the SED fitting process. The sigma clipping is carried out based on the distribution of chi values (sum((D-M)/Derr)).
- **del_wave_nebem** – This parameter defines the Wavelength region (+/- del_wave_nebem) around the emission lines that will be excluded in the fitting of spectral continuum between the model spectrum and the observed one.
- **likelihood** – Choice of likelihood function for the RDSPS method. Only relevant if the fit_method='rdsps'. Options are: 'gauss' for the Gaussian form and 'student_t' for the student's t form.
- **dof** – Degree of freedom (nu) in the Student's t likelihood function. Only relevant if the fit_method='rdsps' and likelihood='student_t'.
- **nwalkers** – Number of walkers in the MCMC process. This parameter is only applicable if fit_method='mcmc'.
- **nsteps** – Number of steps for each walker in the MCMC process. Only relevant if fit_method='mcmc'.
- **nsteps_cut** – Number of first steps of each walkers that will be cut when collecting the final sampler chains. Only relevant if fit_method='mcmc' and store_full_samplers=1.

- **nproc** – Number of processors (cores) to be used in the calculation.
- **initfit_nmodels_mcmc** – Number of models to be used in the initial fitting in the MCMC method. Only relevant if fit_method='mcmc'.
- **perc_chi2** – A percentile in the set of models sorted based on the chi-square values that will be considered in the calculation of the best-fit parameters (i.e., posterior-weighted averages) in the RDSPS fitting. This parameter is only relevant if fit_method='rdsps'.
- **cosmo** – Choices for the cosmology. Options are: (a)'flat_LCDM' or 0, (b)'WMAP5' or 1, (c)'WMAP7' or 2, (d)'WMAP9' or 3, (e)'Planck13' or 4, (f)'Planck15' or 5. These options are similar to the choices available in the Astropy Cosmology package.
- **H0** – The Hubble constant at z=0. Only relevant when cosmo='flat_LCDM' is chosen.
- **Om0** – The Omega matter at z=0.0. Only relevant when cosmo='flat_LCDM' is chosen.
- **store_full_samplers** – Flag indicating whether full sampler models will be stored into the output FITS file or not. Options are: 1 or True for storing the full samplers and 0 or False otherwise.
- **name_out_fits** – Name of the output FITS file. This parameter is optional. If None, a default name will be adopted.

1.21 piXedfit_analysis

piXedfit.piXedfit_analysis.**plot_SED**(*name_sampler_fits*, *logscale_x=False*, *logscale_y=True*, *xrange=None*, *yrange=None*, *wunit='micron'*, *funit='erg/s/cm2/A'*, *decompose=0*, *xticks=None*, *photo_color='red'*, *residual_range=[-1.0, 1.0]*, *show_original_spec=False*, *fontsize_tick=18*, *fontsize_label=25*, *show_legend=True*, *loc_legend=4*, *fontsize_legend=18*, *markersize=100*, *lw=2.0*, *name_plot=None*)

Function for plotting best-fit (i.e., median posterior) model SED from a result of SED fitting.

Parameters

- **name_sampler_fits** – Name of input FITS file containing result of an SED fitting.
- **logscale_x** – Choice for plotting x-axis in logarithmic (True) or linear scale (False).
- **logscale_y** – Choice for plotting y-axis in logarithmic (True) or linear scale (False).
- **xrange** – Range in x-axis. The accepted format is: [xmin,xmax]. If xrange=None, the range will be defined based on the wavelength range of the observed SED.
- **yrange** – Range in y-axis. The accepted format is: [ymin,ymax]. If yrange=None, the range will be defined based on the fluxes range of the observed SED.
- **wunit** – Wavelength unit. Options are: 0 or 'angstrom' for Angstrom unit and 1 or 'micron' for micron unit.
- **funit** – Flux unit. Options are: 0 or 'erg/s/cm2/A', 1 or 'erg/s/cm2', and 2 or 'Jy'.
- **decompose** – Choice for showing best-fit (i.e., median posterior) model SED broken down into its components (1 or True) or just its total (0 or False).
- **xticks** – xticks in list format.
- **photo_color** – Color of photometric fluxes data points. The accepted colors are those available in the matplotlib.

- **residual_range** – Residuals between observed SED and the median posterior model SED. The residual is defined as (D - M)/D, where D represents observed SED, while M is model SED.
- **show_original_spec** – (default=False) Show original best-fit model spectrum before rescaling with polnomial correction. This is only relevant if the data is spectrophotometric.
- **fontsize_tick** – Fontsize for the ticks.
- **fontsize_label** – Fontsize for the labels in the x and y axes.
- **show_legend** – Option for showing a legend.
- **loc_legend** – Location of the legend.
- **fontsize_legend** – Fontsize of the legend.
- **markersize** – Size for the markers of the observed and model SEDs.
- **lw** – (optional, default: 1) Line width for the best-fit (i.e., median posterior) model SED.
- **name_plot** – Desired name for the output plot. This is optional. If None, a default name will be used.

piXedfit.piXedfit_analysis.**plot_corner**(*name_sampler_fits*, *params=['log_sfr', 'log_mass', 'log_dustmass', 'log_fagn', 'log_fagn_bol', 'log_tauagn', 'log_qpah', 'log_umin', 'log_gamma', 'dust1', 'dust2', 'dust_index', 'log_mw_age', 'log_age', 'log_t0', 'log_alpha', 'log_beta', 'log_tau', 'logzsol', 'z', 'gas_logu', 'gas_logz']*, *label_params={'dust1': '$\\hat \\tau_{1}$', 'dust2': '$\\hat \\tau_{2}$', 'dust_index': 'n', 'gas_logu': 'log(U)', 'gas_logz': 'log($Z_{gas}/Z_{\\odot}$)', 'log_age': 'log($\\mathrm{age}_{\\mathrm{sys}}$)', 'log_alpha': 'log($\\alpha$)', 'log_beta': 'log($\\beta$)', 'log_dustmass': 'log(M_{dust})', 'log_fagn': 'log($f_{AGN,*}$)', 'log_fagn_bol': 'log($f_{AGN,bol}$)', 'log_gamma': 'log($\\gamma_{e}$)', 'log_mass': 'log(M_{*})', 'log_mw_age': 'log($\\mathrm{age}_{\\mathrm{M}}$)', 'log_qpah': 'log($Q_{PAH}$)', 'log_sfr': 'log(SFR)', 'log_t0': 'log(t_{0})', 'log_tau': 'log($\\tau$)', 'log_tauagn': 'log($\\tau_{AGN}$)', 'log_umin': 'log(U_{min})', 'logzsol': 'log($Z/Z_{\\odot}$)', 'z': 'z'}*, *params_ranges={'dust1': [0.0, 4.0], 'dust2': [0.0, 4.0], 'dust_index': [-2.2, 0.4], 'gas_logu': [-4.0, -1.0], 'gas_logz': [-2.0, 0.2], 'log_age': [-3.0, 1.14], 'log_alpha': [-2.5, 2.5], 'log_beta': [-2.5, 2.5], 'log_dustmass': [-99.0, -99.0], 'log_fagn': [-5.0, 0.48], 'log_fagn_bol': [-99.0, -99.0], 'log_gamma': [-3.0, -0.824], 'log_mass': [-99.0, -99.0], 'log_mw_age': [-99.0, -99.0], 'log_qpah': [-1.0, 0.845], 'log_sfr': [-99.0, -99.0], 'log_t0': [-2.0, 1.14], 'log_tau': [-2.5, 1.5], 'log_tauagn': [0.7, 2.18], 'log_umin': [-1.0, 1.176], 'logzsol': [-2.0, 0.5], 'z': [-99.0, -99.0]}*, *factor=1.0*, *nbins=12*, *fontsize_label=20*, *fontsize_tick=14*, *name_plot=None*)

Function for producing corner plot that shows 1D and joint 2D posterior probability distributions from a fitting result with the MCMC method.

Parameters

- **name_sampler_fits** – Name of input FITS file containing result of an SED fitting.

- **params** – List of parameters to be shown in the corner plot. This is optional. If default input is used, all the parameters incolved in the SED fitting will be shown in the corner plot.
- **label_params** – Labels for the parameters. The accepted format is a python dictionary.
- **params_ranges** – Desired ranges for the parameters.
- **factor** – Multiplication factor to be applied to stellar mass, SFR, and dust mass.
- **nbins** – Number of binning in the parameter space when calculating the joint posteriors.
- **fontsize_label** – Fontsize of labels in the x and y axes.
- **fontsize_tick** – Fontsize for the ticks.
- **name_plot** – (optional, default: None) Desired name for the output plot.

Returns name_plot
Desired name for the output plot. This is optional. If None, a default name will be used.

piXedfit.piXedfit_analysis.**plot_sfh_mcmc**(*name_sampler_fits*, *nchains=200*, *del_t=0.05*, *lbacktime_max=None*, *yrange=None*, *factor=1.0*, *loc_legend=2*, *fontsize_tick=18*, *fontsize_label=25*, *fontsize_legend=26*, *logscale_x=False*, *logscale_y=False*, *name_plot=None*)

Function for producing SFH plot from a fitting result with the MCMC method. This is only applicable for fitting result that stores the full sampler chains, which is when we set store_full_samplers=1 in the SED fitting functions.

Parameters

- **name_sampler_fits** – Name of input FITS file containing result of an SED fitting.
- **nchains** – Number of randomly chosen sampler chains (from the full samplers stored in the FITS file) to be used for calculating the inferred SFH.
- **del_t** – Width of the look back time binning in unit of Gyr for sampling the star formation history (SFH).
- **lbacktime_max** – Maximum look-back time in the SFH plot. If None, the maximum look-back time is defined from the age of universe at the redshift of the galaxy.
- **yrange** – Range in the y-axis.
- **factor** – Multiplication factor to be applied to the SFH.
- **loc_legend** – Where to locate the legend. This is the same as in the *matplotlib*.
- **fontsize_tick** – Fontsize for the ticks.
- **fontsize_label** – Fontsize of the labels in the x and y axes.
- **fontsize_legend** – Fontsize of the legend.
- **logscale_x** – Choice for plotting x-axis in logarithmic (True) or linear scale (False).
- **logscale_y** – Choice for plotting y-axis in logarithmic (True) or linear scale (False).

Returns name_plot
Desired name for the output plot. This is optional. If None, a default name will be used.

Returns grid_lbt
Look-back times.

Return grid_sfr_p16
16th percentile of the SFR(t).

Return grid_sfr_p50
50th percentile of the SFR(t).

Return grid_sfr_p84
84th percentile of the SFR(t).

1.22 Utils

`piXedfit.utils.filtering.`**`add_filter`**(*filter_name*, *filter_wave*, *filter_transmission*, *filter_cwave*)

Function for adding a new filter transmission function into piXedfit

Parameters

- **`filter_name`** – A given name (in string) for the filter curve
- **`filter_wave`** – Array of wavelength in the filter transmission function
- **`filter_transmission`** – Array of transmission corresponding with the filter_wave
- **`filter_cwave`** – The central wavelength or effective wavelength of the filter

`piXedfit.utils.filtering.`**`change_filter_name`**(*old_filter_name*, *new_filter_name*)

Function for changing a filter name

Parameters

- **`old_filter_name`** – Old filter name.
- **`new_filter_name`** – New filter name.

`piXedfit.utils.filtering.`**`cwave_filters`**(*filters*)

Function for retrieving central wavelengths of a set of filters

Parameters
`filters` – A list of filters names

Returns cwaves
A list of central wavelengths of the filters

`piXedfit.utils.filtering.`**`filtering`**(*wave*, *spec*, *filters*)

Function for convolving a spectrum with a set of filter transmission functions

Parameters

- **`wave`** – array of wavelength of the input spectrum
- **`spec`** – array of fluxes of the input spectrum
- **`filters`** – List of filters name in array of string

Returns fluxes
Array of photometric fluxes

`piXedfit.utils.filtering.`**`get_filter_curve`**(*filter_name*)

Function to get a transmission function of a filter that is available in piXedfit

Parameters
`filter_name` – Name of the filter

Returns wave
Array of wavelength

Returns trans
Array of transmission values

`piXedfit.utils.filtering.list_filters()`

Function for listing the available filters transmission functions in piXedfit

Returns filters
List of filters curves available

`piXedfit.utils.filtering.remove_filter(filter_name)`

Function for removing a filter transmission function from piXedfit

Parameters
filter_name – The filter name.

CHAPTER

TWO

CITATION

If you use this code for your research, please reference Abdurro'uf et al. (2021). If you use the pixel binning module (`piXedfit_bin`), please also reference Abdurro'uf & Akiyama (2017).

CHAPTER

THREE

REFERENCE

A list of a few projects **piXedfit** is benefited from:

- Astropy
- Photutils
- Aniano et al. (2011) who provides convolution kernels for the PSF matching
- FSPS and Python-FSPS stellar population synthesis model
- emcee package for the Markov Chain Monte Carlo (MCMC) Ensemble sampler

PYTHON MODULE INDEX

p

INDEX

O

P

R

S

T

U

V